
Department of

Mathematical Sciences

# Linear Optimisation and Numerical Analysis MX3503

Ian Craw

January 15, 2002, Version 4.2



Additional copies may be obtained from:

Department of Mathematical Sciences
University of Aberdeen
Aberdeen AB9 2TY

DSN: mth202-103713-2

# Foreword

## What the Course Tries to Do

It is always a good idea to start by saying what we are trying to do, and how you will recognise if you have succeeded! Modern education jargon sums up this by emphasising "aims" and "learning outcomes". Here they are in the same form as in the "Catalogue of Courses". It is hard to understand what the outcomes *mean* before you have finished the course; but at that stage, you should come back here and check that they do make sense. If they don't, you may have missed something important.

### Aims

The overall aim of the course is threefold:

- to describe the simplex algorithm and show how it can be used to solve real problems;
- to show how previous results in linear algebra give a framework for understanding the simplex algorithm; and
- to place the simplex algorithm in a more general context by describing other calculus-based and computer based optimisation algorithms.

### Learning Outcomes

By the end of the course the student should be able to:

- formulate mathematically, optimisation problems specified in words;
- recognise a linear programming problem, and in simple cases, to solve it using an appropriate form of the simplex algorithm;
- justify the various steps in the algorithm (ignoring degenerate cases involving cycling) based on the prerequisite linear algebra;
- solve more complicated linear programming problems using MAPLE;
- state, prove and apply in simple cases, the fundamental theorem of duality;
- solve simple matrix games, and describe how the solution derives from duality theory;
- have an overview of nonlinear optimisation problems and be able to describe potential methods of solution, including simulated annealing and calculus based methods;

- discuss genetic algorithms in the context of optimisation, describing choices to be made in the formulation; and

- solve simple non-linear optimisation problems which are amenable to approaches using Lagrange multipliers or the Kuhn Tucker conditions.

## The Course

The simplex algorithm is one of the most successful applications of modern linear algebra. It provides a routine way of solving certain types of maximising or minimising problems which occur frequently in practice. Although it can be reduced to a (relatively) simple algorithm, the emphasis in this course is on describing the reasoning behind the algorithm, to show how it comes out of the linear algebra that was studied last year. We shall not consider directly how it may be programmed efficiently; in practice, if you ever need to use the simplex algorithm, you will have access to a good implementation.

As far as possible, we explore the use of the algorithms using relatively simple examples. However, working manually, it is hard to go much beyond matrices with (say) 4 rows and at most 8 columns without loosing the feel for the problem in a plethora of row operations. Thus for larger problems we use MAPLE, which easily handles larger matrices. We can thus illustrate more than a single feature of the general problem in a given example and give a more rounded or realistic set of examples. In practice we use a relatively simple subset of MAPLE's facilities; Chapter 3 gives a review of all we need. As such this course should be accessible to students *without* previous experience with MAPLE.

Although much of the course is devoted to the Simplex Algorithm and some of its applications, we conclude with a discussion of methods, both traditional and much more modern, which are suited to more general problems. This aims to put the detailed work in an appropriate applications context.

## Syllabus

The following detailed course description was made available in a separate sheet, but it may be convenient to repeat it here.

The course studies linear optimization methods in significant detail before moving on to more general non-linear problems, and will cover most of the following topics.

**Linear Optimization** This occupies roughly three quarters of the course and consists of

- Linear programming problems in practice. Examples including the transport problem. Revision of graphical methods of solution.
- An introduction to, or revison of, the use of MAPLE to manipulate matrices, with particular emphasis on their construction and row-reduction.
- The One Phase Algorithm. A standard form for linear programming problems. Solutions of (redundant) systems of linear equations, basic solutions and degenerate solutions; tableaus and change of basis. The basic solution of a linear programming problem associated with a tableau. The one-phase algorithm, and a partial proof of its correctness. Examples.

- The Two Phase Algorithm. Introduction to the two phase problem. Artificial variables. Examples illustration the various possibilities. A partial proof of the result (ignoring cycling). Linear programming problems without non-negativity restrictions.
- Duality Theory. Primal and dual problems. The bidual. Statement and proof of the fundamental theorem. Examples, and illustrations of the failure of more general versions of duality. Interpretation of duality
- Game theory. An introduction to matrix games and conservative strategies. Pure and mixed strategies. Proof of the fundamental theorem using duality theory. Examples

**Non-linear Optimization** This work is treated more generally and will study topics from

- General introduction to non-linear optimisation. Brief discussion of traditional methods such as steepest descent. Computer based searching including simulated annealing.
- Genetic algorithms: an example of a modern method suitable for a wide variety of problems. A description of the method. Gray coding and its relevance. Methods of generating initial populations; selection mechanisms. Mutation. Examples of use in practice.
- Quasi-linear optimisation methods: how to relax linearity slightly. Lagrange multipliers and the Kuhn - Tucker conditions.

## These Notes

Printed notes are designed to help you get the most from the lectures and accompanying tutorials which form the main part of the course. They contain the material that I use when preparing the actual lectures; in that sense they are *my* lecture notes. They also approximate what you as a student may choose to write down from these lectures; in that sense they are *your* lecture notes. And in each case, they form an approximation: a lecture is a form of communication; I will alter things in the lecture whenever I think a change will improve communication, and you may choose to write down things from the lecture that I have not put in these notes.

> "Lectures were once useful, but now when all can read, and books are so numerous, lectures are unnecessary." *Samuel Johnson, 1799.*

Lecture notes have been around for centuries, either informally, as handwritten notes, or formally as textbooks. Recently improvements in typesetting have made it easier to produce "personalised" printed notes as here, but there has been no fundamental change. Experience shows that very few people are able to use lecture notes as a *substitute* for lectures; if it were otherwise, lecturing, as a profession would have died out by now. To put it another way, "any teacher who can be replaced by a teaching machine, deserves to be". So you should bear in mind that:

- these notes are intended to *complement* the experience you get from attending the lectures; and

- are available to *supplement* the notes that you take in the lectures.

There is significant value in taking your own notes; you are much more likely to see what is going on as you do so. I hope that by having this version available as well, you can quickly correct any errors that occur as you take them.

These notes have a long history; a course rather like this has been given within the Mathematics Department for at least 25 years. During that time many people have taught the course and all have left their mark on it; clarifying points that have proved difficult, selecting the "right" examples and so on. Dr Christopher Clapham was the first to formalise the course as a set of printed notes and this version was developed from them. It is now written in LaTeX which allows a higher level view of the text, and simplifies the preparation of such things as the index on page 132 and numbered equations. You will find that most equations are not numbered, or are numbered symbolically. However sometimes I want to refer back to an equation, and in that case it is numbered within the chapter. Thus Equation (1.1) refers to the first numbered equation in Chapter 1 and so on.

## The Web Version

Printed notes are convenient to use, but have the disadvantage that updating is expensive and inconvenient. In contrast the web is a very recent, but already almost universal medium, which offers convenient and very rapid updating. To take advantage of this, these notes are also available as a set of linked file on the web. Since I can update these files (over 750 when I last counted) easily, this format will always have the most recent version of the notes. They are available at `http://www.maths.abdn.ac.uk/~igc/tch/mx3503/notes/notes.html`.

It is essential to use a graphics-based browser, because each piece of mathematics has to be delivered as a separate image. In the future browsers may be able to cope with mathematics, although I think this unlikely to happen quickly. The conversion from printed notes to HTML is automatic, and has some imperfections. However it *is* getting better; I am now reasonably confident that they say what I intended!

A pdf (Portable Document Format) version is available from the Web. The file can then be viewed, typically in your browser, using Adobe's freely available Acrobat reader. The whole document, or selected pages, can then be printed. This is likely to give much better quality output than printing directly from the web, since the pdf version is based on the original PostScript rather than the derived HTML.

Much of this material lends itself to demonstration, so it is no surprise that there are many web resources available. In the HTML version of this document there are live links to some I have enjoyed. If you come across others, please let me know. And there seems no point in writing out the links in this printed version; they are of no use to you unless you have access to a browser and hence to the web version!

## The MX3503 Mailing List

There is a mailing list associated with this class. You can subscribe to it by sending email to `majordomo@maths.abdn.ac.uk` with a message that has an empty "subject" field and contains the single line `subscribe mx3503-list`. If you add your signature automatically, you can also add the line `end` after the subscribe request, and the signature will be ignored. You are encouraged to join this list. You then mail `mx3503-list@maths.abdn.ac.uk` to contribute to the list.

I have always been happy to deal with questions by email, and have made a point of publishing my email address both on the web and in the printed notes. This list provides a more general way of communicating in which both questions and their answers go to everyone on the list. Here are some reasons why this might be useful:

- It is rare for just one person in a class to have a given problem; by sending the answer to this list, others can benefit from it.
- When a topic causes general concern, the lectures can be changed to cover it in greater detail, or pick up the problem area.
- Often other members of the class can be of more help than the lecturer; this way everyone is aware of the problems and is invited to help or comment.
- I have always been careful to make statements about the examination in front of the whole class – the list provides an equivalent public forum.

Please note that this list is being maintained on the mathematics machines rather than the central University ones, so you need to mail `maths.abdn.ac.uk` to reach it.

Finally some points of netiquette.

- Please remember the usual courtesies; although this is email, it is still fairly public.
- If you send email directly to me, I will not copy it to this list without first getting your permission.
- The list is low security. Although you are technically able to impersonate someone else, please don't do so.
- Keep a copy of the message you get when you join the list to remind you how to leave; you are welcome to leave and re-join as often as you wish.

Sometimes it is useful to pass a message to the whole class. I believe that for most people, this list is more useful way than posting it on a notice board. One natural use would be to cancel a lecture if that has to be done unexpectedly. The message may not reach everyone, but those who read it will be saved the inconvenience of turning up.

Any more questions? Why not mail the list? You won't be the only one with them.

### Computer Algebra Systems

Part of this course involves the use of MAPLE to solve linear programming problems. I'm sure a knowledge of MAPLE will be valuable in your subsequent career; it is probably helpful to say a word about its relevance to this and other mathematics courses. Our aim in general is to teach principles and ideas. Examples are presented and questions set not because the answers themselves are interesting — although sometimes they will be — but with the aim of showing how the principles can be applied. So it is completely pointless to use a packaged routine which gives the answer, unless this helps you to understand the underlying method. You may well get answers using such a package in your subsequent professional career, but you should only do so provided you have a good understanding of the underlying principles.

A computer algebra system is (essentially) infallible, but it can only answer the questions you ask. It is always useful to check you are asking the right questions by being able to do simple cases by hand. You can only do this if you have the sort of understanding we are trying to encourage in this course. You are welcome to use MAPLE to help with tutorial questions whenever you find it useful to do so; in other words, whenever you know exactly what you want to do, but are finding the manipulation heavy.

Examinations will aim to test your understanding of the basic principles, rather than your ability to do long calculations accurately. Only very simple calculators are acceptable as electronic aids, and you should plan your use of MAPLE accordingly; it is a valuable tool, but you should be able to manage without it.

This version of these notes was prepared in my study at home on a Maths Department Sun workstation. The examples included are accurate; I simply arrange for the output file from MAPLE to be processed automatically for inclusion in these notes.

## Books

The recommended book for this course used to be Hartley (1985). It was cheap, well written and centred quite accurately around most of the material in the course. It is now out of print, but there are still copies in various libraries. An even older recommendation is Smythe & Johnson (1966), which has the advantage of being theoretically very thorough. A more modern book, clearly American but very nicely done is Winston (1995). It is not a formal recommendation because it covers much more than is needed, yet misses out a number of the modern methods that probably should be there. Another view is found in Press, Flannery, Teukolsky & Vetterling (1992) which is on the shelves of almost every practising engineer and industrial mathematician. Their view of the simplex algorithm, discussed briefly in Section 1.6, uses less linear algebra than we present here, and provides a nice contrast.

A good modern view of the *standard* methods of looking at non-linear problems can be found in Kaplan (1999). It touches in part on linear optimisation, but its main interest is in the classical non-linear problems. A more standard book is (Kolman & Beck 1995); I would like to recommend this, but it *is* expensive for what is essentially a standard American college textbook.

Press et al. (1992) also gives details about simulated annealing, one of the more modern methods we consider in the second part of the course. The other main one is the use of genetic algorithms, which is becoming increasing popular, and my original reference, (Goldberg 1989), is starting to look dated. The MAPLE "help" system is good, but you may want to read more; the natural place to start is (Char, Geddes, Gonnet, Leong, Monagan & Watt 1992). Throughout, I have quoted the edition of the copy of the book that I use. In general the edition is not critical, although clearly a later edition is more in sympathy with modern ideas about teaching.

## Tutorials

Trying to learn mathematics by *reading* is like learning to play football by watching videos; the real test is how well you can do it yourself! Tutorial sheets will be made available during the course; the exercises there are intended to help you come to grips with the material. Solutions will subsequently be distributed; my aim here is to ensure you spend some time

thinking about problems rather than immediately giving up, and looking at the suggested solutions. This is not however how you are likely to use the web, and so in the web version, exercises are immediately followed by their solution.

There are some examples at the end of many of the Chapters, drawn from previous examination papers. You are strongly advised to try them yourself before looking at the (potential) solutions at the end of the notes. A number of past examination papers are available from the MX3503 homepage. You should make sure you have a proper attempt at each question before looking at the solution. You should be aware that *official* solutions are not made available. Many things can cause the solution to be wrong. One such is when the question is changed but the answer file is not updated. In the end you are responsible for ensuring that you understand the solutions and believe them to be correct. If you have problems in doing so, please seek help.

**Acknowledgements**

Over the years many students have helped to improve these notes by pointing out stupidities, repetitions misprints and so on. Since they have gone on the web, others, often in the USA, have contributed to this gradual improvement by taking the trouble to let me know of difficulties, either in content or presentation. I thank all who have done so, and am happy to receive comments from others.

Ian Craw
Department of Mathematical Sciences
Room 344, Meston Building
email: `Ian.Craw@maths.abdn.ac.uk`
www: `http://www.maths.abdn.ac.uk/~igc`
January 15, 2002

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introducing the Simplex Algorithm

## 1.1 Constrained Optimisation

Almost the whole of this course is concerned with the following general question:

> Find the maximum, or minimum value of a function $f = f(x_1, \ldots, x_n)$ of $n$ real variables subject to the *constraints* that $g_i(x_1, \ldots, x_n) \geq 0$ for $i = 1, \ldots, m$.

Such a problem will be referred to as a **constrained optimisation** problem, and the corresponding value as the **optimum value**. The function we are to optimise is called the **objective function**. The set of vectors $\mathbf{x} = (x_1, \ldots, x_n)$ which satisfy the constraints is known as the **feasible region**. A vector at which the objective function attains its optimal value is known as an **optimal feasible vector**. Of course there is no guarantee that an optimal feasible vector exists for a given problem.

Note first that there is no need to deal with maximising and minimising separately, since finding a minimum of $f(x_1, \ldots, x_n)$ is the same as finding a maximum of $-f(x_1, \ldots, x_n)$. Note also that if we have two constraints $g_1 \geq 0$ and $g_2 \geq 0$ and in addition we know that $g_2 = -g_1$, then $g_1 = 0$; in other words an equality constraint can be reformulated as a pair of "$\geq$" constraints.

As a simple example, consider the problem of maximising $x^2 - 5x + 6$ for $1 \leq x \leq 2$. Such a problem is familiar from first year calculus; it can be written as a constrained optimisation problem by writing $f(x) = x^2 - 5x + 6$ and defining constraints $g_1(x) = x - 1 \geq 0$ and $g_2(x) = 2 - x \geq 0$.

In the first part of the course, we concentrate on a special case of the problem in which both the objective function, and the constraints are *linear* functions. Such a problem is known as a **Linear Programming Problem**, or linear optimisation problem,

## 1.2 Some Sample Problems

We give a number of different examples of problems with linear constraints, aiming to show that a large class of "interesting" problems are of this type.

### 1.2.1 Maximising a Function of Two Variables with Constraints

Our first example can be solved quite simply by geometric methods.

The problem is to maximise $x+y$ subject to the constraints that $2x+y \leq 8$, $x+2y \leq 7$ and $x-y \geq -2$. This is illustrated in Fig. 1.1. The feasible region is shaded, and consists of the portion in the first quadrant (so $x \geq 0$ and $y \geq 0$), which lies below each of the three thick lines. These lines, $y = x+2$, $y+2x = 8$ and $2y+x = 7$ correspond to when the constraint becomes **tight**. The objective function - the thing we are trying to maximise is $x+y$, and the three parallel lines represent lines $x+y = k$ for different values of $k$. The largest value of $k$ is attained at a point within the feasible region when the line just touches the vertex $M$; in the diagram, this corresponds to the largest value of $k$, with increasing values of $k$ corresponding to lines which are further up and to the right.

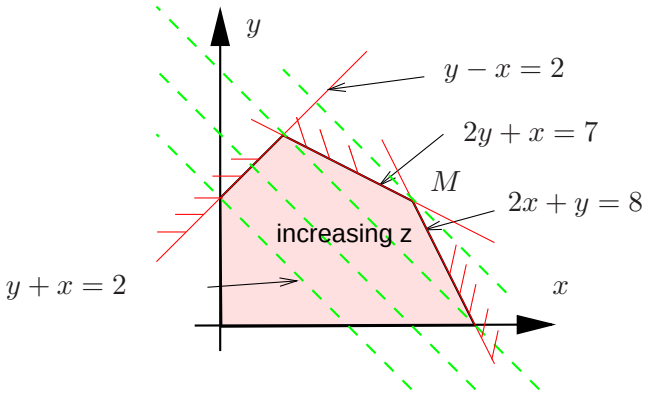


Figure 1.1: Two dimensional optimisation problem.

### 1.2.2 Machine Shop Scheduling

Our next task is to show that problems of this sort can occur in "practical" situations; that problems of interest in the "real world" lead to this type of *constrained optimisation* problem.

A machine shop makes two products called (rather unimaginatively) $A$ and $B$. Product $A$ can be made with two options — as $A_1$ and $A_2$, while product $B$ is available in options $B_1$, $B_2$ and $B_3$. The machine shop makes the two products using an appropriate combination of three machines, which can be used in any order. The production contract requires that 60 units of item $A$ and 85 units of item $B$ be produced per week, although they can be produced in any of the various options. The objective of the exercise is to determine the product mix that is most profitable. The situation is summed up in Table 1.1.

In order to write down in detail what is required, we need to introduce suitable variables.

> *Choosing variables is often the hardest part of the whole process. One way is to think what you need to know in order to solve the problem — give the orders or instruct the foreman. Such variables are often known as* ***decision variables*** *because knowing their values enables a decision to be made. In this case, the decision is "how many of each option of each product do we make each week?"*

It is thus natural to introduce the following variables. Let $x_1$ be the number of units of product $A_1$ to be produced per week, $x_2$ be the number of units of product $A_2$ to be produced per week, $x_3$ be the number of units of product $B_1$ to be produced per week, $x_4$

| Product | Option | Unit production time on machine number | | | Unit Profit |
|---|---|---|---|---|---|
| | | 1 | 2 | 3 | |
| A | 1 | 0.5 | - | 0.2 | 2 |
| | 2 | - | 0.4 | 0.2 | 2.5 |
| B | 1 | 0.4 | 0.3 | - | 5 |
| | 2 | 0.4 | - | 0.3 | 4 |
| | 3 | - | 0.6 | 0.3 | 4 |
| Hours per week that machines are available | | 38 | 31 | 34 | |

Table 1.1: Machine shop costs.

be the number of units of product $B_2$ to be produced per week and $x_5$ be the number of units of product $B_3$ to be produced per week.

The profit from such a product mix is given by

$$P = 2x_1 + 2.5x_2 + 5x_3 + 4x_4 + 4x_5,$$

and this is the function (of $x_1, x_2, \ldots, x_5$) that we wish to maximise. The constraints are of three sorts:

$$\begin{aligned} x_1 + x_2 &= 60, \\ x_3 + x_4 + x_5 &= 85, \end{aligned} \qquad \text{(Required production)}$$

$$\begin{aligned} 0.5x_1 + 0.4x_3 + 0.4x_4 &\leq 38, \\ 0.4x_2 + 0.3x_3 + 0.6x_5 &\leq 31, \\ 0.2x_1 + 0.2x_2 + 0.3x_4 + 0.3x_5 &\leq 34, \end{aligned} \qquad \text{(Machine time)}$$

$$x_i \geq 0 \quad \text{for each } i. \qquad \text{(Reality)}$$

Solving this constrained optimisation problem then gives the values of $x_1, x_2, \ldots, x_5$ which give the most profit for this particular contract.

*1.1. Remark.* Much of the remainder of the course is devoted to solving such problems. When you can, and have enough facility with MAPLE, come back to this problem. You should find that the problem is feasible and that the maximum profit is 520 units.

### 1.2.3 A Transport Problem

In the next introductory example, we give the data in purely symbolic form. As such this describes a *class* of problems, known as **transport problems**.

A firm has warehouses $W_1, W_2, \ldots, W_m$ to supply retail outlets $R_1, R_2, \ldots, R_n$ with a certain product. The warehouse $W_i$ has a supply $s_i$ of the product ($i = 1, \ldots, m$), measured in some convenient units, and we assume that **all** the supply is to be shipped to the retail outlets. In doing this, a demand $d_j$ at outlet $R_j$ **must** be satisfied for each $j = 1, \ldots n$. It is given that the cost of shipping the product from warehouse $W_i$ to retail outlet $R_j$ is proportional to the amount shipped, and that shipping a unit amount costs $c_{ij}$.

We formulate the problem of determining how much of the product should be sent from each warehouse to each retail outlet, so that all demands are satisfied, all the product is shipped, and the transportation cost is minimised.

> *Again we have the problem of choosing variables which enable us to describe a solution. And again they are suggested by the information that you would need to pass to the manager of each warehouse*

Let $x_{ij}$ be the amount of product shipped from warehouse $W_i$ to retail outlet $R_j$. The the total shipping cost is

$$C = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}.$$

Our supply constraint, that we ship all the product from warehouse $W_i$ becomes

$$\sum_{j=1}^{n} x_{ij} = s_i,$$

while the demand constraint, to meet the demand specified at retail outlet $R_j$ is

$$\sum_{i=1}^{m} x_{ij} \geq d_j.$$

Note we have the feasibility constraints that for each pair $(i, j)$, $x_{ij} \geq 0$, since we must ship a non-negative amount of the product.

The problem then becomes one of minimising the cost $C$ subject to these constraints.

### 1.2.4 A Blending Problem

Wine from three European countries is to be blended. We express all costs in £ (perhaps it should be Euros?), so the three wines cost respectively $C_1$, $C_2$ and $C_3$ per litre. The wines are to be blended and sold for $d$ per litre. The wines have acidities $\alpha_1$, $\alpha_2$, and $\alpha_3$, and the blended wine must have an acidity $\leq \alpha$. *Assuming* that acidity blends by volume, so that if the three wines are mixed in the proportions $x_1 : x_2 : x_3$, then, by volume, the mixture has acidity $\alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3$.

*1.2. Remark.* A separate question is whether the assumption is realistic. You could treat this as an excuse to do some practical work if you wish. Even if the assumption is false, there is a great temptation to behave as if it is true; because anything else is *very* much harder to manage. It should come as no surprise to you that most "scientific" assessment procedures behave as though their problem is linear even when it clearly isn't: do *you* think that *all* the marks on a given exam question are equally easy to get? Note that our system invariably assumes that this is the case.

To continue with the problem, the cost per litre of wine which is sold at $d$ is $C_1 x_1 + C_2 x_2 + C_3 x_3$. Our problem is to maximise the profit, which is thus

$$d - \sum_{i=1}^{3} C_i x_i,$$

subject to the constraints that $x_i \geq 0$ and $\alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 \leq \alpha$.

**Another version** This is the same problem, but there are only $Q_i$ litres of each wine available. In this case, we use as variables $q_i$, the number of litres of each wine to be blended. The problem then becomes that of maximising

$$d\sum_{i=1}^{3} q_i - \sum_{i=1}^{3} C_i q_i$$

The limited volume gives the constraint that $q_i \leq Q_i$ for $i = 1, 2, 3$ while the acidity constraint is

$$\alpha_1 \frac{q_1}{\sum_{i=1}^{3} q_i} + \alpha_2 \frac{q_2}{\sum_{i=1}^{3} q_i} + \alpha_3 \frac{q_3}{\sum_{i=1}^{3} q_i} \leq \alpha.$$

As it stands, this last constraint is *not* linear, but can be made so by multiplying through by $\sum_{i=1}^{3} q_i$.

## 1.3 A More Elaborate Example

In the past, the following example has been set as continuous assessment for this class. It is presented here as a relatively realistic example of how the simplex algorithm might be used in practice. You are invited to work the various parts of the example at the appropriate time during the course. At present you can do the "formulation" part. Here then is the "story".

You have been engaged by a manufacturing company because they value your expertise in Linear Programming. This may be a little premature, but they don't need your report until after you have finished Section 3, when you will have the necessary expertise. The company is the Mendip Metals Manufacturing PLC. This example is borrowed, so I have deliberately left in the original location! And the Muchals Metals Manufacturing Company didn't have quite the same foreign ring to it! The details provided by MMM are given in question 1.3 on tutorial sheet 1.

Your eventual aim is to produce a report addressed to the MMM management, advising how much of each of the raw materials to buy, in order to maximise profits. You should also discuss the effects of changes in market conditions. You may in addition offer other advice based on your calculations. At this stage, you should show that the problem of deciding which, if any, of the alloys to manufacture, and from which raw materials, can be expressed as a Linear Programming Problem. Overall, the production of the report is divided into three parts:

- formulating the problem;
- obtaining a solution of the problem in MAPLE; and
- obtaining a useful collection of solutions, and on the basis of your results, writing the report.

There should be no mathematics in the body of the report. The mathematical formulation, and a brief statement of the solution, should go into Appendices, together with any sensitivity results you have calculated. Do *not* go into details of the simplex calculation, which should be done using MAPLE, and not by hand. The entire report will be assessed

by the Board (although not by me!) on the quality and clarity of your recommendations as well as the accuracy of the calculations. I would welcome seeing finished work, and if enough people get that far, will arrange a number of "competing" presentations to the class during the last lecture of the course.

**Some Advice** Here is the advice I gave when the report was to be handed in as CA. You may still find it valuable if you are going to do this exercise "properly".

- Remember than problems often have a number of formulations, and that the first one you think of may not be the simplest. You can expect to spend quite a lot of time setting this problem up.
- The actual calculations should be done in MAPLE, but no intermediate calculations need be shown.
- Your report should be clearly organised. Number the pages. Explain the general structure of your report. Make your recommendations stand out. And think about an executive summary. You may wish to address some of your remarks to specific divisions of the company.
- Legible handwriting or typing is acceptable, but you may also prepare your report in a word processor. If you bind your report, make sure that the writing is still visible, and does not disappear into the binding.
- Ensure that, as consultant, your name appears on the report. It is possible that the board has called for more than one report on the same subject, and you would not wish your work to be confused with that of others.

## 1.4 Linear Programming Problems

With this introduction, we now give a fairly formal definition of the class of problem we are going to study.

**1.3. Definition.** Suppose that one is given a linear (strictly an affine) function of $n$ real variables

$$z = f(x_1, x_2, \ldots, x_n) = c_1x_1 + c_2x_2 + \ldots + c_nx_n + d$$

and a set of linear inequalities and/or equations, called *constraints*

$$\begin{array}{c} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \leq \text{ or } = \text{ or } \geq b_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \leq \text{ or } = \text{ or } \geq b_2, \\ \ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \leq \text{ or } = \text{ or } \geq b_m, \end{array} \tag{1.1}$$

where in each line either $\leq$, $=$ or $\geq$ occurs. The problem of finding $\mathbf{x}$ in $\mathbb{R}^n$, where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, that satisfies the constraints (1.1) and makes $z$ a maximum (or minimum) is called a **Linear Programming Problem** . We saw in Section 1.1 that, if it is convenient, we can always restrict attention to just having "$\leq$" signs, at the expense of having more constraints.

We shall assume that every Linear Programming Problem has included in its constraints, the non-negativity restrictions

$$x_j \geq 0 \quad \text{for } j = 1, 2, \ldots, n. \tag{1.2}$$

and these will be written separately from the other constraints. We will see, in Section 4.5 that this does not in fact limit the class of problems that can be handled by the methods to be discussed. These non-negativity constraints are sometimes known as **reality constraints**. In examples they typically represent quantities that for physical reasons are non-negative.

Any $\mathbf{x}$ satisfying the constraints (1.1) and inequalities (1.2) is called a **feasible solution**. The set of feasible solutions is the **feasible region**. Somewhat perversely, any $\mathbf{x}$ satisfying the constraints (1.1) but not (1.2) is called a **non-feasible solution**.

The function $f$ is called the **objective function** and $z$ the **objective variable**. If $\mathbf{x}$ is a feasible solution that makes $f(x_1, \ldots, x_n)$ a maximum (or minimum) then $x$ is an **optimal** solution and the corresponding value of z is the **optimal value**.

Finally a convention; an element $\mathbf{x}$ in $\mathbb{R}^n$ will be treated as a row vector or as a column vector according to the context.

## 1.5 Standard form

**1.4. Definition.** When written as: maximise (or minimise) $z = x_{n+1}$ subject to

$$\sum_{j=1}^{n+1} a_{ij} x_j = b_i \quad \text{for } i = 1, 2, \ldots, m \qquad \textbf{and } x_j \geq 0 \text{ for } j = 1, 2, \ldots, n. \tag{1.3}$$

a Linear Programming Problem is said to be in **standard form**.

Using matrix notation, this standard form can be written: maximise (or minimise) $x_{n+1}$ subject to $\mathbf{Ax} = \mathbf{b}$, and $x_i \geq 0$ for $i = 1, 2, \ldots, n$, where $\mathbf{x} \in \mathbb{R}^{n+1}$ and $\mathbf{b} \in \mathbb{R}^m$. Notice that in the standard form the objective variable is equal to $x_{n+1}$ and this has no non-negativity restriction; but we will still say that the Linear Programming Problem has non-negativity restrictions. Moreover we write $\mathbf{x} = (x_1, x_2, \ldots, x_n; x_{n+1})$ and if $\mathbf{x}$ satisfies all the conditions in (1.3) we say that $\mathbf{x}$ is a feasible solution. It is not hard to see that, every Linear Programming Problem can be put into standard form in the following sense. For any Linear Programming Problem $P$, there is a Linear Programming Problem $P'$ in standard form, such that:

- a feasible solution of $P$ corresponds to a feasible solution of $P'$, both solutions giving the same objective value, and conversely;
- any optimal solution of $P$ corresponds to an optimal solution of $P'$, both yielding the same optimal value, and conversely; and
- if $P$ has no feasible solution then $P'$ has no feasible solution, and conversely.

The Linear Programming Problem $P'$ is called a *related standard form* for $P$. Rather than prove this formally, we illustrate with an example.

*1.5. Example.* Express the problem of maximising $z = 5x_1 - x_2 + 2x_3 + 4$ subject to

$$\begin{aligned} x_1 - 3x_2 - x_3 &\leq 8 \\ 2x_1 - 2x_3 &\geq 5 \\ x_1 - 6x_2 + x_3 &= -7 \\ x_1, x_2, x_3 &\geq 0. \end{aligned}$$

in standard form.

Let $P$ be the above Linear Programming Problem. Then a related standard form $P'$ for $P$ can be obtained by introducing variables $x_4$ and $x_5$ in the two inequalities and writing $z = x_6$, to obtain: maximise $x_6$ subject to

$$\begin{aligned} x_1 - 3x_2 - x_3 + x_4 &= 8 \\ 2x_1 - 2x_3 - x_5 &= 5 \\ x_1 - 6x_2 + x_3 &= -7 \\ -5x_1 + x_2 - 2x_3 + x_6 &= 4 \\ x_1, x_2, x_3, x_4, x_5 &\geq 0. \end{aligned}$$

The variables, here $x_4$ and $x_5$, introduced into the inequalities to obtain a related standard form, are called **slack variables**.

## 1.6 A Preview

Our aim in the course is to use known linear algebra theory to put the simplex algorithm into a firm theoretical context. To this end we shall use the above transformation to related standard form to work with matrices as our fundamental object rather than inequalities or equations; such an abstraction has significant gains. However there is a perfectly valid *description* of the algorithm in terms of equations which we now describe. You may find it gives you an interesting alternative view of the whole process.

### 1.6.1 An Example

We illustrate the general case with an example, taken from Press et al. (1992, Section 10.8, Page 433). This reference provides a good description of the whole process to the stage of deriving an efficient *algorithm* for doing the computations. In my view the description gets very close to the title of their book — it is primarily designed as a recipe; but their explanation does proceed in terms of equations. The problem is:

> Maximise $z = 2x_1 - 4x_2$ subject to the reality constraints $x_1 \geq 0$, $x_2 \geq 0$, $x_3 \geq 0$, $x_4 \geq 0$ and
>
> $$\begin{aligned} x_3 &= 2 - 6x_1 + x_2, \\ x_4 &= 8 + 3x_1 - 4x_2. \end{aligned}$$

This problem is in a rather special form; we start by analysing it. Note that the variables in the problem fall into two classes; those that appear on the right hand side of the equations,

namely $x_1$ and $x_2$, and the other variables ($z$, $x_3$ and $x_4$) which appear on the left hand side of the equation. We shall, in an obvious way, call these the **right hand variables** and the **left hand variables** respectively. So far there is nothing special. Note that each left hand variable, including the objective function, $z$, appears in only one equation, and that each equation involves a single left hand variable. Finally note that all the constants in the constraints are positive, and that each left hand variable has coefficient +1. An objective function and a set of "equality" constraints in this form is said to be in **restricted normal form**.

Note that since each constraint has an associated left hand variable, there are at least as many variables as constraints — there are actually more variables than constraints if there are *any* right hand variables. Our experience solving linear systems suggest that this is the "usual" situation in which we can expect a solution; had there been more constraints than variables we would have expected an "over-determined" system which is in general inconsistent.

We are familiar with a system in this form when solving a set of equations using Gaussian elimination. Having diagonalised the left hand side of the system as much as possible, it is easy to solve it — simply move the remaining variables to the right hand side, give them arbitrary values, and solve for each of the left hand variables. In our case, we obtain a *feasible* solution to the optimisation problem by setting each right hand side variable to 0, and then solving for the left hand side variables. Since the constant in each constraint equation is non-negative, the complete solution is necessarily feasible. In our case we have the solution

$$x_1 = 0, \quad x_2 = 0; \qquad x_3 = 2, \quad x_4 = 8.$$

With these values, the objective function has the value $z = 0$. This wasn't special; in the same way, we can guarantee to write down a feasible solution to any problem in restricted normal form.

### 1.6.2 Improving the Solution

Can we find a larger value of the objective function than 0? Since we had some freedom in assigning the values of $x_1$ and $x_2$, can we do better?[1] Looking again at the objective function, namely $z = 2x_1 - 4x_2$, it is clear there is no gain from trying to increase $x_2$, since that will only decrease $z$. In contrast increasing $x_1$ may be useful. One way to do that is to arrange for $x_1$ to be a left hand side variable. We do so by rearranging the first constraint:

$$x_3 = 2 - 6x_1 + x_2 \quad \text{so} \quad x_1 = \tfrac{1}{3} - \tfrac{1}{6}x_3 + \tfrac{1}{6}x_2.$$

We can now return our system to restricted normal form by using this transformed equation giving $x_1$ as a left hand variable to substitute for $x_1$ whenever it occurs as a right hand variable. We have

$$\begin{aligned} z = 2x_1 - 4x_2 \quad &\text{so} \quad z = 2[\tfrac{1}{3} - \tfrac{1}{6}x_3 + \tfrac{1}{6}x_2] - 4x_2 = \tfrac{2}{3} - \tfrac{1}{3}x_3 - \tfrac{11}{3}x_2, \\ x_4 = 8 + 3x_1 - 4x_2 \quad &\text{so} \quad x_4 = 8 + 3[\tfrac{1}{3} - \tfrac{1}{6}x_3 + \tfrac{1}{6}x_2] - 4x_2 = 9 - \tfrac{1}{2}x_3 - \tfrac{7}{2}x_2. \end{aligned}$$

[1] We didn't have *complete* freedom in assigning $x_1$ and $x_2$; each had to be non-negative, but we also had to choose them so the left hand variables came out to be non-negative.

Our optimisation problem is now back in restricted normal form since it turns out that each constraint equation has a non-negative constant So we can obtain a feasible solution as before; this time we have:

$$x_3 = 0, \quad x_2 = 0; \qquad x_1 = \tfrac{1}{3}, \quad x_4 = 9.$$

With these values, the objective function has the value $z = \frac{2}{3}$; the change of left hand variables has resulted in an *improved* value for $z$.

We now have

$$z = \tfrac{2}{3} - \tfrac{1}{3}x_3 - \tfrac{11}{3}x_2,$$

and since $x_3 \geq 0$ and $x_2 \geq 0$, necessarily $z \leq \frac{2}{3}$. Clearly then we have found the maximum value of $z$ and so finished the problem.

This looks like a very special method, being applied to a very special version of the general Linear Programming Problem. However we saw in Section 1.5 that introducing slack variables can easily generate what we have just called left hand variables. Consider the problem of maximising $2x_1 - 4x_2$ subject to the constraints that

$$\begin{aligned} 6x_1 - x_2 &\leq 2, \\ -3x_1 + 4x_2 &\leq 8. \end{aligned}$$

Introducing $x_3$ and $x_4$ as slack variables in the first and second inequalities respectively gives the problem we have just discussed, and it is already in restricted normal form! Our aim in what follows is to show that

- all Linear Programming Problems can be reduced to restricted normal form;
- if a maximum exists and hasn't been found, the objective function can be improved by swapping right hand and left hand variables as above; and
- if a maximum exists, it will be found after a finite number of steps.

In fact we have no more use for the ideas of left and right hand variables, or of restricted normal form, except when discussing this formulation of the problem.

*1.6. Example.* We used the first constraint to make $x_1$ a left hand variable. Show that had we used the *second* constraint to make $x_1$ a left hand variable, thus making $x_4$ a right hand variable instead of $x_3$, the corresponding solution would *not* have been feasible. Strictly, the resulting system would *not* have been in restricted normal form.

## 1.7 Questions 1 (Hints and solutions start on page 107.)

*1.1. Q.* A paper mill produces rolls of paper each of which is 18 ft. wide. These are then cut into various widths as required by customers. A roll can be cut into narrower rolls as many times as necessary.

a) A customer requires rolls in widths of 9 ft., 7ft. and 5 ft. In how may different ways can the 18 ft. rolls be cut to yield (one or more of) these sizes? In each case, say how much waste is created.

b) An order is received for 10 rolls 9 ft. wide, 20 rolls 7 ft. wide and 50 rolls 5 ft. wide. Formulate a linear programming problem to fill the order using the minimum number of rolls. Do *not* solve this problem, but say what difficulties you see in putting the solution into practice.

c) Now suppose that surplus rolls less than 5 ft. wide are sold for $£k$ per ft., so that a roll of width $w < 5$ is sold for $£kw$. Suppose also that a standard roll costs $£P$ to produce, and that each cut costs $£C$. Formulate a linear programming problem to satisfy the requirements at minimum net cost.

*1.2. Q.* A factory makes three types of small decorative garden sculpture, known as Bashful, Dozy and Happy. Their manufacture is done using three different machines, called $A$, $B$ and $C$, which can be used in any order. The number of hours needed on each of the three machines to make each sculpture is given in the following table

| | $A$ | $B$ | $C$ | Profit/unit |
|---|---|---|---|---|
| Bashful | 2 | 1 | 2 | £2 |
| Dozy | 1 | 0 | 3 | £4 |
| Happy | 0 | 3 | 2 | £3 |

which also gives the net profit made on each of the sculptures. Machine $A$ is available for 43 hours per week, machine $B$ for 37 and machine $C$ for 42 hours per week.

The problem is to decide how many of each type should be made each week in order to maximise the profit on the operation. Formulate the problem as a linear programming problem. [You are *not* asked to solve the problem.]

*1.3. Q.* A pharmaceutical company is creating a tablet for a new drug. Each tablet is to contain a binder, a disintegrant and a filler in addition to the active drug ingredient, which is to be 14% of the weight of each tablet. Chemical and physical considerations mean that the weight of the disintegrant should not exceed 25% of the combined weights of the binder and the active ingredient, and that there should be at most 10 times as much filler as binder. The disintegrant costs £15, the binder £50 and filler £2 per kilogram.

The problem is to decide how to formulate the tablet in order to minimise its cost. Express the problem as a linear programming problem. [You are *not* asked to solve the problem.]

*1.4. Q.* A manufacturer makes a range of three types of car, $A$, $B$ and $C$ in two factories; an engine plant $E$ and a body factory $F$. The nett profit on a car of type $A$ is £1100, on type $B$ it is £1200 and on type $C$ it is £1450. There are 10120 labour units (a labour unit is approximately one "man hour") available in $E$ and 11000 labour units in factory $F$ each month and the number of labour units, needed to build the various products is given in Table 1.2.

| Car type | Engine | Body |
|---|---|---|
| A | 8 | 8 |
| B | 8 | 9 |
| C | 9 | 11 |

Table 1.2: Productions times

Formulate the problem of maximising the profit subject to these constraints as a linear programming problem. [You are *not* asked to solve it.]

One purported solution of this problem shows no cars of type $B$ are made. Is this plausible? Describe briefly three ways in which this mathematical model of car production might be modified to be more realistic.

*1.5. Q.* A Natural Food store makes up three types of muesli, which it sells as "Crunchy", Healthy" and "Rich". Each type is made by mixing different proportions of cereals, nuts and dried fruit. Existing advertising material means that the restrictions and selling prices given in Table 1.3 are fixed.

| Type | Cereal | Dried Fruit | Nuts | Selling price per kilo (£) |
|---|---|---|---|---|
| Crunchy | | | At least 60% | 1.60 |
| Healthy | At least 60% | | At most 20% | 1.20 |
| Rich | At most 20% | At least 60% | | 2.00 |

Table 1.3: Required muesli ingredients and selling prices.

The suppliers can deliver at most 100 kilos of cereal at £0.80 per kilo, 80 kilos of dried fruit at £1.50 per kilo and 60 kilos of nuts at £1.00 per kilo each week. The store is able to sell all the muesli that it mixes. Formulate the problem of finding the most profitable mixing scheme as a linear programming problem. You are *not* asked to solve the problem.

# Chapter 2

# The One-phase Simplex Algorithm

## 2.1 Solutions of Systems of Linear Equations

From here until Section 2.6, we shall forget about the objective function and just consider the constraints.

For a Linear Programming Problem in standard form, these are just a set of equations $\mathbf{Ax} = \mathbf{b}$, and so we are studying such systems of linear equations. The $m \times n$ inhomogeneous system $\mathbf{Ax} = \mathbf{b}$ can be written as

$$x_1\mathbf{a}_1 + x_2\mathbf{a}_2 + \ldots + x_n\mathbf{a}_n = \mathbf{b}$$

where $\mathbf{a}_j$ is the j-th column of the matrix $\mathbf{A}$. Thus any solution of the equations is essentially an expression of $\mathbf{b}$ as a linear combination of the columns of $\mathbf{A}$.

**2.1. Definition.** The **column space** of $\mathbf{A}$, denoted by $M_A$, is the subspace of $\mathbb{R}^m$ spanned by the columns of $\mathbf{A}$.

There are just three possibilities:

1. the column $\mathbf{b} \notin M_A$ and so there are no solutions;
2. the column $\mathbf{b} \in M_A$ and $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n$ are linearly independent, in which case there is a unique solution $\mathbf{x}$; or
3. the column $\mathbf{b} \in M_A$ and $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n$ are linearly dependent, in which case there are infinitely many solutions.

Clearly case (3) is the only case that is of interest for Linear Programming Problems, in contrast to our usual interest in uniqueness. Different solutions of the set of equations correspond to possibly different values of the objective function; and as such we can ask which of the solutions gives the *best* answer.

## 2.2 Basic Solutions

We assume that $\mathbf{b} \in M_A$ and that $\mathbf{a}_1$,$\mathbf{a}_2$, ... ,$\mathbf{a}_n$ are linearly dependent.

Let $B = \{\mathbf{a}_{h_1}, \mathbf{a}_{h_2}, \ldots, \mathbf{a}_{h_k}\}$, where $1 \leq h_i \leq n$ for $i = 1, 2, \ldots, k$. Suppose that $B$ is a basis for $M_A$, so that $k = \dim M_A$.

**2.2. Definition.** The solution **associated** with the basis $B$ is the unique vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ such that $\mathbf{b} = \sum_{j=1}^{n} x_j \mathbf{a}_j$ with $x_j = 0$ if $\mathbf{a}_j \notin B$.

A solution $\mathbf{x}$ is **basic** if it is the solution associated with some basis of $M_A$, chosen from the columns of $\mathbf{A}$. If $\mathbf{a}_j$ belongs to the basis under consideration, we call $\mathbf{a}_j$ a **basic column** and the corresponding $x_j$ a **basic variable**.

**2.3. Lemma.** *The following hold for basic solutions of* $\mathbf{Ax} = \mathbf{b}$*:*

- *the number of basic solutions of* $\mathbf{Ax} = \mathbf{b}$ *is finite; and*
- *a solution* $\mathbf{x}$ *is basic if and only if the set* $\{\mathbf{a}_j : x_j \neq 0\}$ *is linearly independent.*

*Proof.* We get a basic solution for each basis $B$, so we must count the number of bases for $M_A$ consisting of columns of $A$. Assuming that $\dim A = k$, we can have no more than

$$ {}_nC_k = \frac{n!}{k!(n-k)!}, $$

since this is the number of ways of picking *any* $k$ columns from the possible $n$ columns from $A$. It may be that some of the sets of $k$ columns of $A$ are not linearly independent, in which case we get fewer bases, but in any case, the number is finite.

If $\mathbf{x}$ is a basic solution, we can write $\mathbf{x} = \sum x_j \mathbf{a}_j$ in terms of all the columns of $\mathbf{A}$. There is some basis $B$ consisting of columns of $A$ such that $x_j = 0$ if $\mathbf{a}_j \notin B$. Thus

$$ B \supseteq \{\mathbf{a}_j \mid x_j \neq 0\}, $$

and the right hand side, being a subset of a basis, is certainly linearly independent.

Conversely if $\{\mathbf{a}_j \mid x_j \neq 0\}$ is linearly independent, we can extend it to get a basis $B$ of $M_A$ consisting of columns of $\mathbf{A}$. Clearly $\mathbf{x}$ is then basic. □

**2.4. Definition.** A basic solution $\mathbf{x}$ is **degenerate** if $\{\mathbf{a}_j : x_j \neq 0\}$ is a *proper* subset of a basis for $M_A$.

Thus if a degenerate basic solution $\mathbf{x}$ is associated with a basis $B$ then at least one of the basic variables is zero — and we generate $\mathbf{x}$ without using as many basic columns as might have been expected.

**2.5. Definition.** A **basic feasible solution** is a solution that is both basic and feasible.

## 2.3 Row-equivalent matrices

We say that the matrices $\mathbf{A}$ and $\mathbf{C}$ are **row-equivalent** if one can be obtained from the other by a succession of operations of the following kind

- interchanging two rows;
- multiplying a row by a non-zero constant; or
- replacing a row by the sum of it and (a non-zero multiple of) another row.

Recall that these operations are known as *row operations* and that they have an interpretation directly in terms of matrix algebra. We write $R(\mathbf{A})$ for the matrix obtained by performing the row operation $R$ on the matrix $\mathbf{A}$. Each row operation $R$ defines an *elementary matrix* $\mathbf{E}_R = R(\mathbf{I}_n)$ by performing the operation $R$ on the identity matrix $\mathbf{I}_n$. A calculation then shows that $\mathbf{E}_R \cdot \mathbf{A} = R(\mathbf{A})$. Recall also, or check directly, that each $\mathbf{E}_R$ is invertible. Thus if $\mathbf{A}$ and $\mathbf{C}$ are row-equivalent matrices, there is a sequence $\mathbf{E}_1$, $\mathbf{E}_2$, ..., $\mathbf{E}_k$ of elementary matrices such that $\mathbf{C} = \mathbf{E}_1\mathbf{E}_2 \dots \mathbf{E}_k\mathbf{A}$, and so, in particular, an invertible matrix $\mathbf{P}$ such that $\mathbf{C} = \mathbf{PA}$.

Let $[\mathbf{A}|\mathbf{b}]$ be the augmented matrix associated with $\mathbf{Ax} = \mathbf{b}$. Note that the column space $M_A$ associated with the matrix $\mathbf{A}$ is the same as the column space of $[\mathbf{A}|\mathbf{b}]$, precisely because of our assumption above that $\mathbf{b} \in M_A$.

**2.6. Theorem.** *If* $[\mathbf{A}|\mathbf{b}]$ *and* $[\mathbf{C}|\mathbf{d}]$ *are row-equivalent then* $\mathbf{Ax} = \mathbf{b}$ *and* $\mathbf{Cx} = \mathbf{d}$ *have the same set of solutions.*

*Proof.* Since $[\mathbf{A}|\mathbf{b}]$ and $[\mathbf{C}|\mathbf{d}]$ are row-equivalent, there is an invertible matrix $\mathbf{P}$ such that $\mathbf{PA} = \mathbf{C}$, and $\mathbf{Pb} = \mathbf{d}$. Then

$$\begin{aligned} \mathbf{Ax} = \mathbf{b} &\Leftrightarrow \mathbf{PAx} = \mathbf{Pb} \quad \text{because } \mathbf{P} \text{ is invertible} \\ &\Leftrightarrow \mathbf{Cx} = \mathbf{d}. \end{aligned}$$

□

**2.7. Corollary.** *If* $\mathbf{A}$ *and* $\mathbf{C}$ *are row-equivalent then* $\mathbf{Ax} = \mathbf{0}$ *and* $\mathbf{Cx} = \mathbf{0}$ *have the same set of solutions.*

*Proof.* If $\mathbf{A}$ and $\mathbf{C}$ are row equivalent, then $[\mathbf{A}|\mathbf{0}]$ is row equivalent to $[\mathbf{C}|\mathbf{0}]$, since $\mathbf{P0} = \mathbf{0}$ for any invertible $\mathbf{P}$. The result thus follows from 2.6. □

**2.8. Corollary.** *Let* $\mathbf{A} = [\mathbf{a}_1, \dots, \mathbf{a}_n]$ *and* $\mathbf{C} = [\mathbf{c}_1, \dots, \mathbf{c}_n]$ *be* $m \times n$ *row-equivalent matrices. Then*

$$\sum_{i=1}^{n} x_i\mathbf{a}_i = 0 \quad \textit{if and only if} \quad \sum_{i=1}^{n} x_i\mathbf{c}_i = 0.$$

*Proof.* This is just a restatement of the previous result. □

**2.9. Corollary.** *Let* $\{\mathbf{a}_{h_1}, \dots, \mathbf{a}_{h_k}\}$ *be a set of columns of* $\mathbf{A}$ *and let the corresponding set of columns of* $\mathbf{C}$ *be* $\{\mathbf{c}_{h_1}, \dots, \mathbf{c}_{h_k}\}$. *Assume that* $\mathbf{A}$ *and* $\mathbf{C}$ *are row-equivalent. Then*

1. $\mathbf{a}_j = \sum_{i=1}^{k} \lambda_i\mathbf{a}_{h_i}$ *if and only if* $\mathbf{c}_j = \sum_{i=1}^{k} \lambda_i\mathbf{c}_{h_i}$;
2. *The set* $\{\mathbf{a}_{h_1} \dots, \mathbf{a}_{h_k}\}$ *is a basis for* $M_A$ *if and only if* $\{\mathbf{c}_{h_1} \dots, \mathbf{c}_{h_k}\}$ *is a basis for* $M_C$.
3. $\dim M_A = \dim M_C$.

## 2.4 Computation of basic solutions

**2.10. Definition.** Let $\mathbf{e}_k$ be the $m$-dimensional unit vector $(0,\dots,1,\dots,0)$ with a 1 in the $k$-th place and 0's elsewhere. The $m \times n$ matrix $\mathbf{A}$ is in **canonical form** if all the vectors $\mathbf{e}_k$ occur as columns of $\mathbf{A}$, except that $\mathbf{e}_k$ need not occur if the $k$-th row of $\mathbf{A}$ is zero. We say $[\mathbf{A}|\mathbf{b}]$ is in canonical form if $\mathbf{A}$ is.

Now suppose that solutions to $\mathbf{Ax} = \mathbf{b}$ do exist. Then each matrix $[\mathbf{C}|\mathbf{d}]$ that is row-equivalent to $[\mathbf{A}|\mathbf{b}]$ and is in canonical form determines, using 2.9, a basic solution for $\mathbf{Ax} = \mathbf{b}$. We say that this is the basic solution associated with $[\mathbf{C}|\mathbf{d}]$. We show how this can occur with an example.

*2.11. Example.* Let

$$[\mathbf{A}|\mathbf{b}] = \left(\begin{array}{cccc|c} 6 & -16 & -3 & 1 & 13 \\ -7 & 21 & 4 & -1 & -10 \\ -3 & 9 & 2 & -1 & -2 \end{array}\right) \quad \text{and} \quad [\mathbf{C}|\mathbf{d}] = \left(\begin{array}{cccc|c} 0 & 2 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 & 7 \\ 0 & 4 & 1 & 0 & 10 \end{array}\right).$$

Given that $[\mathbf{C}|\mathbf{d}]$ is row-equivalent to $[\mathbf{A}|\mathbf{b}]$, interpret Corollary 2.9 by finding a basic solution of $\mathbf{Ax} = \mathbf{b}$.

*Solution* Note first that $[\mathbf{C}|\mathbf{d}]$ is in canonical form; columns 1, 3 and 4 are respectively $\mathbf{e}_2$, $\mathbf{e}_3$ and $\mathbf{e}_1$. We have $\mathbf{c}_1 = \mathbf{e}_2$, $\mathbf{c}_3 = \mathbf{e}_3$ and $\mathbf{c}_4 = \mathbf{e}_1$. Clearly

$$\mathbf{d} = \begin{pmatrix} 1 \\ 7 \\ 10 \end{pmatrix} = 7\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + 10\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} + 1\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = 7\mathbf{c}_1 + 10\mathbf{c}_3 + 1\mathbf{c}_4.$$

Hence $\mathbf{b} = 7\mathbf{a}_1 + 10\mathbf{a}_3 + 1\mathbf{a}_4$, from 2.9(1). Explicitly, we have

$$\mathbf{b} = \begin{pmatrix} 13 \\ -10 \\ -2 \end{pmatrix} = 7\begin{pmatrix} 6 \\ -7 \\ 3 \end{pmatrix} + 10\begin{pmatrix} -3 \\ 4 \\ 2 \end{pmatrix} + 1\begin{pmatrix} 1 \\ -1 \\ -1 \end{pmatrix} = 7\mathbf{a}_1 + 10\mathbf{a}_3 + 1\mathbf{a}_4.$$

Since $\{\mathbf{c}_1, \mathbf{c}_3, \mathbf{c}_4\}$ is a basis for $M_C$, $\{\mathbf{a}_1, \mathbf{a}_3, \mathbf{a}_4\}$ is a basis for $M_A$, by 2.9(2). Thus we obtain $\mathbf{x} = (7, 0, 10, 1)$, a basic solution for $\mathbf{Ax} = \mathbf{b}$.

From now on we shall often make the implicit assumption that $M_A = \mathbb{R}^m$. In the notation used earlier, this is the assumption that $k = m$.

**2.12. Definition.** A **tableau** for $\mathbf{Ax} = \mathbf{b}$ is a matrix $[\mathbf{C}|\mathbf{d}]$ in canonical form, row-equivalent to $[\mathbf{A}|\mathbf{b}]$, with, listed down the side, the ordered basis of $M_A$ used to determine the basic solution associated with $[\mathbf{C}|\mathbf{d}]$.

A tableau for the equation $\mathbf{Ax} = \mathbf{b}$ is illustrated in Table 2.1.

Notice that $\mathbf{b} = \sum_{i=1}^{m} u_i \mathbf{a}_{h_i}$, and hence $\mathbf{b} = \sum_{j=1}^{n} x_j \mathbf{a}_j$, where $x_j = u_i$ if $j = h_i$ for some $i = 1,\dots,m$ and $x_j = 0$ otherwise. Notice also that, in a similar way, $\mathbf{a}_j = \sum_{i=1}^{m} y_{ij} \mathbf{a}_{h_i}$, where $y_{ij}$ is the $(i,j)$-th element in the array.

| | $\mathbf{a}_1$ | $\mathbf{a}_2$ | ... | $\mathbf{a}_{h_i}$ | ... | $\mathbf{a}_j$ | ... | $\mathbf{a}_n$ | $\mathbf{b}$ |
|---|---|---|---|---|---|---|---|---|---|
| $\mathbf{a}_{h_1}$ | | | | 0 | | $y_{1j}$ | | | $u_1$ |
| $\mathbf{a}_{h_2}$ | | | | 0 | | $y_{2j}$ | | | $u_2$ |
| $\vdots$ | | | | $\vdots$ | | $\vdots$ | | | $\vdots$ |
| $\mathbf{a}_{h_i}$ | | | | 1 | | $y_{ij}$ | | | $u_i$ |
| $\vdots$ | | | | $\vdots$ | | $\vdots$ | | | $\vdots$ |
| $\mathbf{a}_{h_m}$ | | | | 0 | | $y_{mj}$ | | | $u_m$ |

Table 2.1: A tableau for the equation $\mathbf{Ax} = \mathbf{b}$.

The standard way of reducing a matrix to canonical form is by **pivoting**. Let $\mathbf{P} = [p_{ij}]$. Choose any nonzero entry $p_{rs}$. This is taken as the pivot element and the $r$-th row is the **pivot row**. Now

- multiply the pivot row by $1/p_{rs}$, and
- add $-p_{ks}$ times the (new) pivot row to the $k$-th row, for all $k \neq r$.

This results in a row-equivalent matrix $\mathbf{Q}$ such that the $s$-th column of $\mathbf{Q}$ is $\mathbf{e}_r$ and any vectors $\mathbf{e}_k$ with $k \neq r$ that were present as columns of $\mathbf{P}$ remain as columns of $\mathbf{Q}$. It follows that starting with $[\mathbf{A}|\mathbf{b}]$ and pivoting successively at convenient entries in $\mathbf{A}$ only, a matrix in canonical form, row-equivalent to $[\mathbf{A}|\mathbf{b}]$, is obtained.[1]

*2.13. Example.* Illustrate this process with the set of equations

$$\begin{array}{rcrcrcrcr} 6x_1 & - & 16x_2 & - & 3x_3 & + & x_4 & = & 13 \\ -7x_1 & + & 21x_2 & + & 4x_3 & - & x_4 & = & -10 \\ -3x_1 & + & 9x_2 & + & 2x_3 & - & x_4 & = & -2. \end{array}$$

*Solution* We show the process by which one obtains the basic solution $\mathbf{x} = (7, 0, 10, 1)$ associated with the basis $\mathbf{a}_1, \mathbf{a}_3, \mathbf{a}_4$ in Table 2.2. Here, and elsewhere, the pivot element will be marked with an asterisk.

Note that the final part of Table 2.2 gives us, according to definition 2.12, a tableau for the original set of equations.

## 2.5 Change of basis

In following the simplex algorithm for linear programming, we move from one basic solution to another. This amounts to changing from one basis to another. We do this by changing one basis vector at a time.

**2.14. Definition.** Let $B$ be a basis for a vector space $V$ and $\mathbf{v}$ a vector in $B$. Then a vector $\mathbf{u}$ of $V$ can **replace** $\mathbf{v}$ in B if $B \setminus \{\mathbf{v}\} \bigcup \{\mathbf{u}\}$ is a basis for $V$.

[1] This is one place where we make the implicit assumption that $M_A = \mathbb{R}^m$; otherwise we could not guarantee that *every* $\mathbf{e}_k$ could be produced , as required to get a matrix in canonical form.

| | $\mathbf{a}_1$ | $\mathbf{a}_2$ | $\mathbf{a}_3$ | $\mathbf{a}_4$ | $\mathbf{b}$ |
|---|---|---|---|---|---|
| | 6 | $-16$ | $-3$ | $1^*$ | 13 |
| | $-7$ | 21 | 4 | $-1$ | $-10$ |
| | $-3$ | 9 | 2 | $-1$ | $-2$ |
| $\mathbf{a}_4$ | 6 | $-16$ | $-3$ | 1 | 13 |
| | $-1$ | 5 | 1 | 0 | 3 |
| | 3 | $-7$ | $-1^*$ | 0 | 11 |
| $\mathbf{a}_4$ | $-3$ | 5 | 0 | 1 | $-20$ |
| | $2^*$ | $-2$ | 0 | 0 | 14 |
| $\mathbf{a}_3$ | $-3$ | 7 | 1 | 0 | $-11$ |
| $\mathbf{a}_4$ | 0 | 2 | 0 | 1 | 1 |
| $\mathbf{a}_1$ | 1 | $-1$ | 0 | 0 | 7 |
| $\mathbf{a}_3$ | 0 | 4 | 1 | 0 | 10 |

Table 2.2: Obtaining the basic solution $\mathbf{x} = (7, 0, 10, 1)$ associated with the basis $\mathbf{a}_1, \mathbf{a}_3, \mathbf{a}_4$.

All this is saying is that if we remove $\mathbf{v}$ and put $\mathbf{u}$ in its place, we are interested in whether we still get a basis. The new set of vectors may be linearly independent, in which case we have another basis, or linearly dependent, in which case we may not replace $\mathbf{v}$ by $\mathbf{u}$. Here is a useful criterion to decide when we can make such a change.

**2.15. Theorem.** *The vector* $\mathbf{u}$ *can replace* $\mathbf{v}$ *in* $B$ *if and only if, when* $\mathbf{u}$ *is expressed in terms of* $B$ *the coefficient of* $\mathbf{v}$ *is nonzero.*

*Proof.* Suppose first that when $\mathbf{u}$ is expressed in terms of $B$ the coefficient of $\mathbf{v}$ is nonzero. Since a basis is a spanning set, we can write

$$\mathbf{u} = \lambda\mathbf{v} + \sum_{i=1}^{n} \lambda_i \mathbf{v}_i$$

for some coefficients $\lambda$ and $\lambda_i$, and we are given $\lambda \neq 0$. In order to show that $B' = B \setminus \{\mathbf{v}\} \bigcup \{\mathbf{u}\}$ is a basis, it is enough to show it is linearly independent, since it has the same number of elements as the basis $B$. So assume that

$$\mu\mathbf{u} + \sum_{i=1}^{n} \mu_i \mathbf{v}_i = \mathbf{0};$$

we show that each coefficient vanishes. From the given expression for $\mathbf{u}$ in terms of $B$, we have

$$\mu\lambda\mathbf{v} + \sum_{i=1}^{n} (\mu\lambda_i + \mu_i)\mathbf{v}_i = \mathbf{0},$$

and since this is a linear relation between members of the basis $B$, each coefficient must vanish. In particular, since $\lambda \neq 0$, we have $\mu = 0$, and so also $\mu_i = 0$ for all $i$. Thus the set $B'$ is linearly independent.

Conversely, suppose that the vector $\mathbf{u}$ can replace $\mathbf{v}$ in $B$, and define $B'$ as above. Since it is a basis, it spans the space, so we can write

$$\mathbf{v} = \mu\mathbf{u} + \sum_{i=1}^{n} \mu_i \mathbf{v}_i,$$

and $\mu$ is non-zero, since otherwise we have

$$\mathbf{v} = \sum_{i=1}^{n} \mu_i \mathbf{v}_i,$$

and this is a non-trivial linear combination of (linearly independent) elements from the basis $B$, which cannot occur.

Thus $\mu$ is non-zero, we can write

$$\mathbf{u} = \frac{1}{\mu}\mathbf{v} - \sum_{i=1}^{n} \frac{\mu_i}{\mu} \mathbf{v}_i,$$

and the coefficient $\frac{1}{\mu}$ is non-zero as required. □

Now, in a tableau for $\mathbf{Ax} = \mathbf{b}$, let $\mathbf{a}_{h_r}$ be one of the basic columns and $\mathbf{a}_s$ some column of $A$. Then $\mathbf{a}_s$ can replace $\mathbf{a}_{h_r}$ in the basis $\{\mathbf{a}_{h_1}, \mathbf{a}_{h_2}, \ldots, \mathbf{a}_{h_m}\}$ if and only if $y_{rs} \neq 0$. If $y_{rs} \neq 0$, then $\mathbf{a}_{h_r}$ is replaced by $\mathbf{a}_s$ by pivoting at $y_{rs}$. In other words, if we wish to change the basis so as to include $\mathbf{a}_s$, when choosing a row to swap out of the basis by pivoting, we only consider rows for which the corresponding entry is non - zero, in order to keep a basis.

## 2.6 Return to the Linear Programming Problem

Now let us consider the problem: maximise (or minimise) $x_{n+1}$ subject to $\mathbf{Ax} = \mathbf{b}$, and $x_1, x_2, \ldots, x_n \geq 0$. We shall assume that the final column of $\mathbf{A}$ is $\mathbf{a}_{n+1} = (0, 0, \ldots, 1)$, in view of the way in which the standard form of a Linear Programming Problem is set up. Henceforth $\mathbf{a}_{n+1}$ will also be called $\mathbf{e}$ and we make the rule that, in any tableau T for $\mathbf{Ax} = \mathbf{b}$, the vector $\mathbf{e}$ must be in the basis as the last member of the basis. The tableau T thus takes the form shown in Table 2.3.

The entries in the last column of the tableau give the associated basic solution. The entry $z$ is precisely the value of the objective variable, also denoted by $x_{n+1}$. The associated basic solution is feasible if and only if $u_i \geq 0$ for $i = 1, 2, \ldots, m$.

Suppose it should happen that for some $s \neq h_r$, we have $y_{rs} = 1$ and $y_{is} = 0$ for $i \neq r$. Then replacing $\mathbf{a}_{h_r}$ by $\mathbf{a}_s$ in the basis has essentially no effect. We shall assume that this unlikely occurrence never arises.

**2.16. Theorem.** *Suppose that the basic solution associated with a tableau is feasible. Then replacing $\mathbf{a}_{h_r}$ by $\mathbf{a}_s$ in the basis yields*

1. *a distinct basic feasible solution if and only if $y_{rs} > 0$ and*

$$\frac{u_r}{y_{rs}} = \min_i \left\{ \frac{u_i}{y_{is}} : y_{is} > 0 \right\} \neq 0; \quad \textit{and}$$

| | $\mathbf{a}_1$ | $\mathbf{a}_2$ | ... | $\mathbf{a}_s$ | ... | $\mathbf{a}_n$ | $\mathbf{e}$ | $\mathbf{b}$ |
|---|---|---|---|---|---|---|---|---|
| $\mathbf{a}_{h_1}$ | | | | $y_{1s}$ | | | 0 | $u_1$ |
| $\mathbf{a}_{h_2}$ | | | | $y_{2s}$ | | | 0 | $u_2$ |
| $\vdots$ | | | | $\vdots$ | | | $\vdots$ | $\vdots$ |
| $\mathbf{a}_{h_i}$ | | | | $y_{is}$ | | | 0 | $u_i$ |
| $\vdots$ | | | | $\vdots$ | | | $\vdots$ | $\vdots$ |
| $\mathbf{a}_{h_r}$ | | | | $y_{rs}$ | | | 0 | $u_r$ |
| $\vdots$ | | | | $\vdots$ | | | $\vdots$ | $\vdots$ |
| $\mathbf{a}_{h_m}$ | | | | $y_{ms}$ | | | 0 | $u_m$ |
| $\mathbf{e}$ | $v_1$ | $v_2$ | ... | $v_s$ | ... | $v_n$ | 1 | $z$ |

Table 2.3: More detail in a tableau for the equation $\mathbf{Ax} = \mathbf{b}$.

*2. the same basic feasible solution if and only if $y_{rs} \neq 0$ and $u_r = 0$.*

*Proof.* In order to replace $\mathbf{a}_{h_r}$ by $\mathbf{a}_s$ in the basis, we pivot at $y_{rs}$ in the tableau. This is shown in Table 2.4, where we do the pivot in two steps for simplicity.

We start with a basic feasible solution. In order to ensure that we keep a *basic* solution at each stage of the pivot, it is essential by Theorem 2.15 that $y_{rs} \neq 0$. We now examine what is needed in order to ensure we have a new basic *feasible* solution.

Note first that if $u_r = 0$, then although we have moved to a new basic feasible solution, it is not distinct from the old one. For the old basic feasible solution has $x_{h_i} = u_i$ for all $i$, and $x_i = 0$ whenever $\mathbf{a}_i$ is not in the basis. In particular, $x_{h_r} = 0$ since $u_r = 0$ and $x_s = 0$, since $\mathbf{a}_s$ is not in the basis. After replacing $\mathbf{a}_{h_r}$ by $\mathbf{a}_s$ in the basis, the same result still holds, and none of the other parts of the basic solution have changed. We thus assume in what follows that $u_r \neq 0$.

Clearly we must have $u_r/y_{rs} \geq 0$ for feasibility, otherwise the requirement that $x_s \geq 0$ fails. Since we started with a feasible solution, $u_r > 0$, this then means that $y_{rs} > 0$ is a necessary condition for the new basic solution to be feasible. The other requirement for feasibility is that the remaining members of the $\mathbf{b}$ column are non - negative, so

$$u_i - y_{is}(u_r/y_{rs}) \geq 0 \quad \text{for } i = 1, 2, \ldots m,\ i \neq r.$$

If $y_{is} \leq 0$, this condition certainly holds, since $u_r/y_{rs} \geq 0$, so we are increasing $u_i$ when we make the change; thus the solution stays feasible.

It remains to consider the situation when $y_{is} > 0$, in which case, we must have

$$u_i - y_{is}(u_r/y_{rs}) \geq 0, \qquad \text{or} \qquad \frac{u_i}{y_{is}} \geq \frac{u_r}{y_{rs}},$$

and this must hold for each relevant $i$. Note also that the division performed to get the second form of the above condition is valid, since we are assuming that $y_{is} > 0$. Thus, in order that the new solution should remain feasible, the minimum value of $u_i/y_{is}$, taken over all indices for which $y_{is} > 0$, occurs when $i = r$. □

| | $\mathbf{a}_1$ | $\mathbf{a}_2$ | ... | $\mathbf{a}_s$ | ... | $\mathbf{a}_n$ | $\mathbf{e}$ | $\mathbf{b}$ |
|---|---|---|---|---|---|---|---|---|
| $\mathbf{a}_{h_1}$ | | | | $y_{1s}$ | | | 0 | $u_1$ |
| $\mathbf{a}_{h_2}$ | | | | $y_{2s}$ | | | 0 | $u_2$ |
| $\vdots$ | | | | $\vdots$ | | | $\vdots$ | $\vdots$ |
| $\mathbf{a}_{h_i}$ | | | | $y_{is}$ | | | 0 | $u_i$ |
| $\vdots$ | | | | $\vdots$ | | | $\vdots$ | $\vdots$ |
| $\mathbf{a}_{h_r}$ | | | | 1 | | | 0 | $u_r/y_{rs}$ |
| $\vdots$ | | | | $\vdots$ | | | $\vdots$ | $\vdots$ |
| $\mathbf{a}_{h_m}$ | | | | $y_{ms}$ | | | 0 | $u_m$ |
| $\mathbf{e}$ | $v_1$ | $v_2$ | ... | $v_s$ | ... | $v_n$ | 1 | $z$ |
| $\mathbf{a}_{h_1}$ | | | | 0 | | | 0 | $u_1 - y_{1s}(u_r/y_{rs})$ |
| $\mathbf{a}_{h_2}$ | | | | 0 | | | 0 | $u_2 - y_{2s}(u_r/y_{rs})$ |
| $\vdots$ | | | | $\vdots$ | | | $\vdots$ | $\vdots$ |
| $\mathbf{a}_{h_i}$ | | | | 0 | | | 0 | $u_i - y_{is}(u_r/y_{rs})$ |
| $\vdots$ | | | | $\vdots$ | | | $\vdots$ | $\vdots$ |
| $\mathbf{a}_s$ | | | | 1 | | | 0 | $u_r/y_{rs}$ |
| $\vdots$ | | | | $\vdots$ | | | $\vdots$ | $\vdots$ |
| $\mathbf{a}_{h_m}$ | | | | 0 | | | 0 | $u_m - y_{ms}(u_r/y_{rs})$ |
| $\mathbf{e}$ | $v_1$ | $v_2$ | ... | 0 | ... | $v_n$ | 1 | $z - v_s(u_r/y_{rs})$ |

Table 2.4: Moving from one basic column to a new one.

Suppose that it has been decided which new column $\mathbf{a}_s$ should be included in the basis. Then part (i) is important for it determines which basis element $\mathbf{a}_{h_r}$ should be replaced. Part (ii) states that on pivoting at $y_{rs}$ the same basic feasible solution is obtained only if the original basic feasible solution was degenerate.

*2.17. Example.* Minimise $-4x_1 + 3x_2$ subject to

$$\begin{aligned} x_1 - 2x_2 &\geq -4, \\ 2x_1 + 3x_2 &\leq 13, \\ x_1 - x_2 &\leq 4, \\ x_1, x_2 &\geq 0. \end{aligned}$$

*Solution* Table 2.5 shows three tableaus for this problem.

The first $T_1$ is obtained from the related standard form and has basic feasible solution $(0, 0, 4, 13, 4; 0)$. Pivoting about the position $y_{31}(= 1)$ gives tableau $T_2$ with basic feasible solution $(4, 0, 8, 5, 0; -16)$. Pivoting about the position $y_{22}(= 5)$ then gives tableau $T_3$ with basic feasible solution $(5, 1, 7, 0, 0; -17)$. The asterisks indicate the pivot elements.

We now turn to the problem of deciding which new column $\mathbf{a}_s$ should be included in the basis.

**2.18. Definition.** The entry $v_s$ in the bottom row of a tableau has the **proper sign for improvement** if either $v_s < 0$ in a *maximising* problem, or $v_s > 0$ in a *minimising* problem.

| | | $\mathbf{a}_1$ | $\mathbf{a}_2$ | $\mathbf{a}_3$ | $\mathbf{a}_4$ | $\mathbf{a}_5$ | $\mathbf{e}$ | $\mathbf{b}$ |
|---|---|---|---|---|---|---|---|---|
| | $\mathbf{a}_3$ | $-1$ | 2 | 1 | 0 | 0 | 0 | 4 |
| $T_1$ | $\mathbf{a}_4$ | 2 | 3 | 0 | 1 | 0 | 0 | 13 |
| | $\mathbf{a}_5$ | $1^*$ | $-1$ | 0 | 0 | 1 | 0 | 4 |
| | $\mathbf{e}$ | 4 | $-3$ | 0 | 0 | 0 | 1 | 0 |
| | $\mathbf{a}_3$ | 0 | 1 | 1 | 0 | 1 | 0 | 8 |
| $T_2$ | $\mathbf{a}_4$ | 0 | $5^*$ | 0 | 1 | $-2$ | 0 | 5 |
| | $\mathbf{a}_1$ | 1 | $-1$ | 0 | 0 | 1 | 0 | 4 |
| | $\mathbf{e}$ | 0 | 1 | 0 | 0 | $-4$ | 1 | $-16$ |
| | $\mathbf{a}_3$ | 0 | 0 | 1 | $-1/5$ | $7/5$ | 0 | 7 |
| $T_3$ | $\mathbf{a}_2$ | 0 | 1 | 0 | $1/5$ | $-2/5$ | 0 | 1 |
| | $\mathbf{a}_1$ | 1 | 0 | 0 | $1/5$ | $3/5$ | 0 | 5 |
| | $\mathbf{e}$ | 0 | 0 | 0 | $-1/5$ | $-18/5$ | 1 | $-17$ |

Table 2.5: Three tableaus for Example 2.17.

**2.19. Theorem.** *Suppose that, in a tableau with corresponding objective value $z$, the entry $v_s$ has the proper sign for improvement and that on pivoting at $y_{rs}$ a new basic feasible solution is obtained with objective value $z'$. Then in a maximising problem we have $z' > z$, and in a minimising problem we have $z' < z$.*

*Proof.* We refer back to the second tableau in Table 2.4, and note that the value in the cell in the bottom right of the table is the value $z$ of the objective function, because $\mathbf{e}$ remains in the basis of the column space. Writing $z'$ for the new value of the objective function, and noting that we always have $u_r/y_{rs} \geq 0$, we have

$$z' \leq z \text{ iff } v_s > 0, \text{ while } z' \geq z \text{ if } v_s > 0.$$

Further, if we assume we are moving to a new *distinct* basic feasible solution, so can assume that $u_r/y_{rs} > 0$, then

$$z' < z \text{ iff } v_s > 0, \text{ while } z' > z \text{ if } v_s > 0.$$

□

This theorem ensures that we choose a column that improves the objective value. We are now in a position to describe the so-called 'one-phase' simplex algorithm.

## 2.7 The One-phase Simplex Algorithm

Suppose that T is a tableau such that the associated basic solution is feasible. (It may be necessary to use the second phase of the algorithm, described below in Section 4.2, to obtain such a tableau.) Choose, if possible, an $s$ such that $v_s$ has the proper sign for improvement. Then choose, if possible, an $r$ such that another (not necessarily new) basic feasible solution is obtained on replacing $\mathbf{a}_{h_r}$ by $\mathbf{a}_s$ in the basis by pivoting about $y_{rs}$. Continue the process as long as possible.

Note that we *never* try to pivot in the same column twice, since after a pivot, the corresponding $v_s$ no longer has a proper sign for improvement. Thus even if we don't get a

new basic feasible solution, the algorithm does not stall, by repeating the *same* choice each time.

In this description, there are two occurrences of "if possible". The process is bound to terminate at one of these points. If there is no $v_s$ with the proper sign for improvement then an optimal solution has been obtained. If on the other hand it is not possible to find a suitable $r$ then the problem has no optimal solution. At this stage, we have not yet proved these claims; we consider them further in Section 2.8, where we establish that *provided* the process terminates, it does so as claimed here. Table 2.1 gives a flow-chart that illustrates and amplifies the description just given.

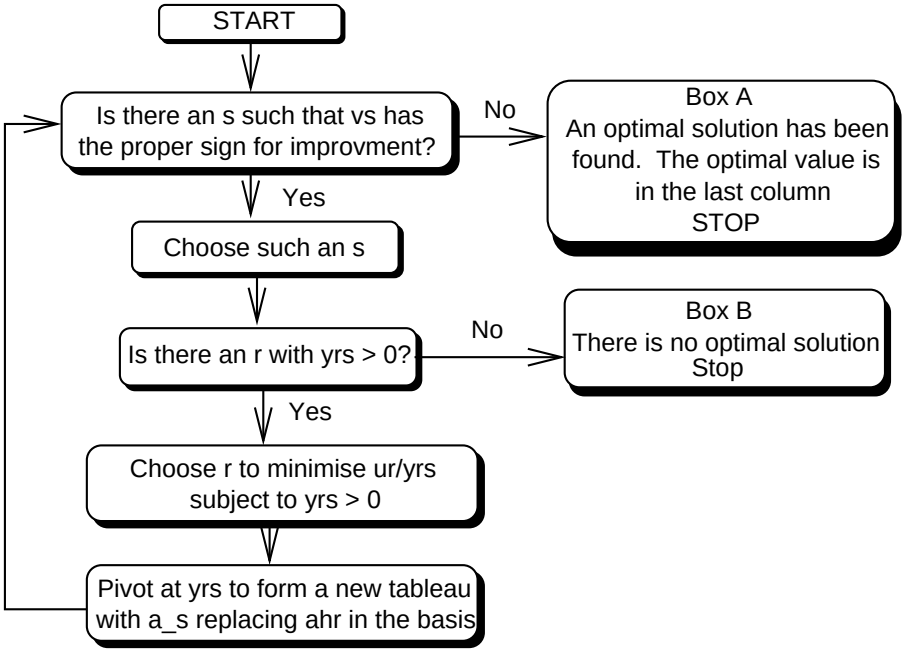


Figure 2.1: Flow chart for one-phase simplex algorithm.

*2.20. Example.* Maximise $4x_1 + 3x_2$ subject to

$$\begin{aligned} 3x_1 + 4x_2 &\leq 12, \\ 7x_1 + 2x_2 &\leq 14, \\ x_1, x_2 &\geq 0. \end{aligned}$$

*Solution* Table 2.6 shows the tableaus obtained by following the algorithm. The tableau $T_1$ is obtained from the related standard form and has basic feasible solution $(0, 0, 12, 14; 0)$. Pivoting about $y_{12}(= 4)$ gives tableau $T_2$ with basic feasible solution $(0, 3, 0, 8; 9)$. Pivoting about $y_{21}(= 22/4)$ then gives $T_3$ with basic feasible solution $(16/11, 21/11, 0, 0, 127/11)$. There is now no $v_s$ with the proper sign for improvement. We have an optimal solution of the original problem $x_1 = 16/11$, $x_2 = 21/11$ with optimal value $127/11$.

| | | $\mathbf{a}_1$ | $\mathbf{a}_2$ | $\mathbf{a}_3$ | $\mathbf{a}_4$ | $\mathbf{e}$ | $\mathbf{b}$ |
|---|---|---|---|---|---|---|---|
| | $\mathbf{a}_3$ | 3 | $4*$ | 1 | 0 | 0 | 12 |
| $T_1$ | $\mathbf{a}_4$ | 7 | 2 | 0 | 1 | 0 | 14 |
| | $\mathbf{e}$ | $-4$ | $-3$ | 0 | 0 | 1 | 0 |
| | $\mathbf{a}_2$ | 3/4 | 1 | 1/4 | 0 | 0 | 3 |
| $T_2$ | $\mathbf{a}_4$ | $22^*/4$ | 0 | $-2/4$ | 1 | 0 | 8 |
| | $\mathbf{e}$ | $-7/4$ | 0 | 3/4 | 0 | 1 | 9 |
| | $\mathbf{a}_2$ | 0 | 1 | 7/22 | $-3/22$ | 0 | 21/11 |
| $T_3$ | $\mathbf{a}_1$ | 1 | 0 | $-1/11$ | 2/11 | 0 | 16/11 |
| | $\mathbf{e}$ | 0 | 0 | 13/22 | 7/22 | 1 | 127/11 |

Table 2.6: Example 2.20: three tableaus.

*2.21. Example.* Minimise $-4x_1 + 3x_2$ subject to

$$\begin{aligned} x_1 - 2x_2 &\geq -4, \\ 2x_1 + 3x_2 &\leq 13, \\ x_1 - x_2 &\leq 4, \\ x_1, x_2 &\geq 0. \end{aligned}$$

This example was studied in Section 2.6 and the tableaus given there in Table 2.5 were obtained by following the algorithm. In $T_3$, of Table 2.5 no $v_s$ has the proper sign for improvement, so an optimal solution of the original problem is $x_1 = 5$, $x_2 = 1$ with optimal value $-17$.

## 2.8 A partial proof

To be sure that the description given is a genuine algorithm, it is necessary to fill in three gaps. We must prove that

1. the claim relating to Box A is true;
2. the claim relating to Box B is true; and
3. the process is bound to terminate after a finite number of steps.

We can deal completely with (1) and (2) but will see that difficulties arise in trying to establish (3).

**2.22. Theorem.** *If in a tableau there is no $v_s$ with the proper sign for improvement then the corresponding basic feasible solution is optimal.*

*Proof.* Suppose for clarity that the original problem is a maximising problem, and that our final tableau exhibits a basic feasible solution $\hat{\mathbf{x}} = (\hat{x}_1, \ldots, \hat{x}_n; \hat{x}_{n+1})$. Assume also that none of the $v_j$'s exhibits the proper sign for improvement, so they are all positive, and our algorithm has halted in Box A. Note that, since we have a basic feasible solution, we have $\hat{x}_{n+1} = z$.

This tableau is row-equivalent to the original problem, and thus represents a system of equations with the same solutions as the initial set of equations. Suppose we have any other feasible solution $\mathbf{x}$ of the initial system $\mathbf{Ax} = \mathbf{b}$ — not necessarily a basic one. In particular, $\mathbf{x}$ gives a solution of the last equation in the tableau, and so

$$\sum_{j=1}^{n} v_j x_j + x_{n+1} = z \qquad \text{and so} \qquad \hat{x}_{n+1} - x_{n+1} = \sum_{j=1}^{n} v_j x_j.$$

By our assumption, all the $v_j$'s are positive, and each $x_j \geq 0$, because this is a feasible solution. Thus $\hat{x}_{n+1} \geq x_{n+1}$, and so $\hat{x}_{n+1} = z$ is the optimal solution. □

This establishes (1), the claim relating to Box A.

**2.23. Theorem.** *Suppose that in a tableau there is an $s$ such that $v_s$ has the proper sign for improvement and $y_{rs} < 0$ for all $r = 1, 2, \ldots, m$. Then there is no optimal solution.*

*Proof.* We use linearity crucially here. Specifically, we use the fact that adding any solution of the equation $\mathbf{Ax} = \mathbf{0}$ to a solution of the equation $\mathbf{Ax} = \mathbf{b}$ gives another solution of the equation $\mathbf{Ax} = \mathbf{b}$.

Assume again for clarity that we are solving a maximising problem, that we have found a column such that $v_s \leq 0$, but that for every $r$, we have $y_{rs} < 0$. Then since this column expresses $\mathbf{a}_s$ in terms of the current basis of the column space, we have

$$\mathbf{a}_s = \sum_{i=1}^{m} y_{is} \mathbf{a}_{h_i} + v_s \mathbf{e}.$$

Re-arranging gives

$$\mathbf{a}_s + \sum_{i=1}^{m} (-y_{is}) \mathbf{a}_{h_i} + (-v_s)\mathbf{e} = \mathbf{0}. \tag{2.1}$$

Note that in this form of the equation, *all* the coefficients are positive.

Now let the current basic feasible solution be $(x_1, x_2, \ldots, x_n; z)$, so that

$$\sum_{i=1}^{m} x_i \mathbf{a}_{h_i} + z\mathbf{e} = \mathbf{b}. \tag{2.2}$$

Adding a multiple $\vartheta$ of equation (2.1) to equation (2.2) then gives

$$\theta \mathbf{a}_s + \sum_{i=1}^{m} (x_i - \vartheta y_{is}) \mathbf{a}_{h_i} + (z - \vartheta v_s)\mathbf{e} = \mathbf{b}$$

and this gives a family of feasible solutions with objective values $z' = z - \vartheta v_s$ for every $\vartheta > 0$. □

This establishes (2), the claim relating to Box B. In these circumstances there is no optimal solution in that the feasible region is unbounded and an arbitrarily large (positive if maximising, negative if minimising) objective value is possible. Indeed, as we have shown, one can find a family $x(\theta)$ of feasible solutions such that, as $\theta \to \infty$, the objective value $f(x(\theta))$ tends to $+\infty$ (if maximising) or $-\infty$ (if minimising).

*2.24. Example.* Maximise $2x_1 + x_2$ subject to

$$\begin{aligned} -x_1 + x_2 &\leq 1, \\ x_1 - 2x_2 &\leq 2, \\ x_1, x_2 &\geq 0. \end{aligned}$$

Table 2.7 shows the tableaus obtained by following the algorithm. The tableau $T_1$ is obtained from the related standard form and has basic feasible solution $(0, 0, 1, 2; 0)$. Pivoting about $y_{21}(= 1)$ gives tableau $T_2$ with basic feasible solution $(2, 0, 3, 0; 4)$. Now $v_2(= -5)$ has the proper sign for improvement but there is no $r$ such that $y_{r2} > 0$. Thus there is no optimal solution.

| | | $\mathbf{a}_1$ | $\mathbf{a}_2$ | $\mathbf{a}_3$ | $\mathbf{a}_4$ | **e** | **b** |
|---|---|---|---|---|---|---|---|
| $T_1$ | $\mathbf{a}_3$ | $-1$ | $1$ | $1$ | $0$ | $0$ | $1$ |
| | $\mathbf{a}_4$ | $1^*$ | $-2$ | $0$ | $1$ | $0$ | $2$ |
| | **e** | $-2$ | $-1$ | $0$ | $0$ | $1$ | $0$ |
| $T_2$ | $\mathbf{a}_3$ | $0$ | $-1$ | $1$ | $1$ | $0$ | $3$ |
| | $\mathbf{a}_1$ | $1$ | $-2$ | $0$ | $1$ | $0$ | $2$ |
| | **e** | $0$ | $-5$ | $0$ | $2$ | $1$ | $4$ |

Table 2.7: Tableaus showing that Example 2.24 has no optimal solution.

From the method used in the proof of the preceding theorem, we find that

$$(2 + 2\theta, \theta, 3 + \theta, 0; 4 + 5\theta)$$

is a feasible solution for all $\theta > 0$. Thus $x_1 = 2 + 2\theta$, $x_2 = \theta$ is a feasible solution of the original problem with objective value $4 + 5\theta$. These form a family $x(\theta)$ of the kind described above.

**2.25. Theorem.** *If every basic feasible solution of a Linear Programming Problem is non-degenerate then the process of following the simplex method does terminate after a finite number of steps.*

*Proof.* It is enough to show that our algorithm terminates. However, except in the case of degeneracy, we move to a different basic feasible solution each time, and cannot return to one already visited, since the objective function strictly improves. But, by Lemma 2.3, there are only a finite number of distinct bases for the column space, so termination must occur after this many steps. □

When there is degeneracy, there is the possibility that the simplex method will return to a previous basic feasible solution and then go round and round in an endless repetition, a process known as cycling. It is said to be a fact, however, that although degeneracy is quite common in practice, cycling has occurred only in artificially constructed examples; however there is a claim that one such problem occurs in the solution of a practical queueing model (Kolman & Beck 1995, Page 127).

*2.26. Example.* Maximise $\frac{3}{4}x_1 - 20x_2 + \frac{1}{2}x_3 - 6x_4$ subject to

$$\begin{aligned} \tfrac{1}{4}x_1 - 8x_2 - x_3 + 9x_4 + x_5 &= 0 \\ \tfrac{1}{2}x_1 - 12x_2 - \tfrac{1}{2}x_3 + 3x_4 + x_6 &= 0 \\ x_3 + x_7 &= 1 \\ x_1, x_2, \ldots, x_7 &\geq 0. \end{aligned}$$

*Solution* It is easy but tedious[2] to check that cycling occurs if the following sequence of bases is taken: $\{\mathbf{a}_5, \mathbf{a}_6, \mathbf{a}_7\}$, $\{\mathbf{a}_1, \mathbf{a}_6, \mathbf{a}_7\}$, $\{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_7\}$, $\{\mathbf{a}_3, \mathbf{a}_2, \mathbf{a}_7\}$, $\{\mathbf{a}_3, \mathbf{a}_4, \mathbf{a}_7\}$, $\{\mathbf{a}_5, \mathbf{a}_4, \mathbf{a}_7\}$, $\{\mathbf{a}_5, \mathbf{a}_6, \mathbf{a}_7\}$. However, by taking the sequence $\{\mathbf{a}_5, \mathbf{a}_6, \mathbf{a}_7\}$, $\{\mathbf{a}_5, \mathbf{a}_1, \mathbf{a}_7\}$, $\{\mathbf{a}_5, \mathbf{a}_1, \mathbf{a}_3\}$, the process can be made to terminate in the optimal basic feasible solution $(1, 0, 1, 0, \frac{3}{4}, 0, 0; \frac{3}{4})$. We show how to check this by machine in Section 3.4

The following result, though positive, is of only theoretical interest:

**2.27. Theorem.** *It is always possible, starting from any given basis with an associated basic feasible solution, to choose a sequence of bases so that the one-phase simplex method terminates.*

In practice something called **Bland's Rule** (Kolman & Beck 1995) can implement the above and so avoid the possibility of cycling. It can be shown that if the choices made during the algorithm are done in an ordered way, then cycling is avoided. The rule is as follows:-

- when there is a choice of pivot column, always choose the one with the lowest index — the one furthest to the left; and
- when there is a choice of pivot row, always prefer the row labelled by the basic variable with the smallest subscript.

## 2.9 Questions 2 (Hints and solutions start on page 109.)

*2.1. Q.* Use the Simplex Algorithm to maximise $5x_1 + 2x_2$, subject to $x_1, x_2 \geq 0$ and

$$\begin{aligned} -x_1 + x_2 &\geq -2, \\ 2x_1 + x_2 &\leq 7. \end{aligned}$$

Either give the maximum value of the objective function, and values of $x_1$ and $x_2$ at which this maximum is obtained, or explain briefly why there is no maximum.

*2.2. Q.* Use the Simplex Algorithm to maximise $-3x_1 + 4x_2$, subject to $x_1, x_2 \geq 0$ and

$$\begin{aligned} 2x_1 - x_2 &\geq -5, \\ x_1 + 3x_2 &\leq 22. \end{aligned}$$

Either give the maximum value of the objective function, and values of $x_1$ and $x_2$ at which this maximum is obtained, or explain briefly why there is no maximum.

[2] Thus an ideal place to use MAPLE.

*2.3. Q.* Use the Simplex Algorithm to maximise $3x_1 - x_2$, subject to $x_1, x_2 \geq 0$ and

$$\begin{aligned} -x_1 &+ 2x_2 \geq -5, \\ 2x_1 &+ x_2 \leq 15. \end{aligned}$$

Either give the maximum value of the objective function, and values of $x_1$ and $x_2$ at which this maximum is obtained, or explain briefly why there is no maximum.

*2.4. Q.* Use the Simplex Algorithm to minimise $-x_1 - 3x_2 + x_3$, subject to $x_1, x_2, x_3 \geq 0$ and

$$\begin{aligned} 3x_1 - 2x_2 + 3x_3 &\leq 3, \\ 2x_1 - x_2 - 6x_3 &\geq -5. \end{aligned}$$

Either give the minimum value of the objective function and the values of $x_1$, $x_2$ and $x_3$ at which this minimum is obtained, or explain briefly why there is no minimum.

*2.5. Q.* Let $V$ be a finite dimensional vector space with basis $\{\mathbf{v}_1, \mathbf{v}_2 \ldots, \mathbf{v}_n\}$. Given $\mathbf{u} \in V$,

$$\mathbf{u} = \lambda_1 \mathbf{v}_1 + \lambda_2 \mathbf{v}_2 + \cdots + \lambda_n \mathbf{v}_n \quad \text{say,}$$

state a necessary and sufficient condition on the coefficient $\lambda_1$ that $\{\mathbf{u}, \mathbf{v}_2 \ldots, \mathbf{v}_n\}$ is a basis for $V$. Prove that your statement is correct.

# Chapter 3

# Using Maple

In this Chapter we have two rather different aims. First we show how the MAPLE package can be used to check a lot of the algebra that you will be doing. It should thus provide a rather easy introduction to symbolic algebra in general, and bring immediate rewards if your pivoting is not as accurate as it might be. The second aim is to give an alternative view of the simplex algorithm, which may make its relationship to equation solving clearer. MAPLE uses this second view of the process, and although it is significantly longer to write out than our standard version of the algorithm, another view is often helpful. Finally we use the package to verify the result on cycling described in Example 2.26, something that is rather tedious to check by hand.

We describe briefly the procedures necessary to run a Maple session. Much of this will be familiar to anyone who has used the PC classrooms already. Even if you are familiar with the logging on sequence, you should still read through this as there are certain specific details which are required to get Maple working.

## 3.1 Getting Access to Maple

In Aberdeen MAPLE runs both on PC Windows operating systems and on Unix executing on the machine known as **sysa**. These instructions are for using the Unix version of Maple on sysa connecting from a PC via the **X Windows** system.

### 3.1.1 Using Maple in a PC Classroom

We give brief instructions for starting the Maple application, applicable to most of the PC classrooms.[1] There are two steps to the process; getting access to the Unix machine "sysa", and then running MAPLE on sysa. In this section we describe how to get access to sysa.

1. Log-on, select the '`Science & Engineering`' folder by double clicking it (it should be on the left hand side of the screen) and then select the '`Mathematical Sciences`' folder.

[1] In previous years it has been possible to use MAPLE in the class library. This is not so at present, but we hope it will be available again next year.

2. Select the yellow `X` icon labelled **exceed** to start `X-Windows`. You will find an old Maple icon next to this; please don't try to use MAPLE by selecting it; that version of the software is out of date.
3. If you are given a choice of servers, select `sysa` to manage your X-windows session;
4. You will now have a `sysa` login dialog displayed. The next step is to run MAPLE as described in Section 3.1.2.
5. **Logging out.** Remember the need to log out before you leave the PC. After quitting Maple, select the 'start' button (lower left of the screen), select 'shut down', then select 'close all programs ... ' and click OK.

You can also get access to sysa from the `Common Applications` folder going via `Internet Options`. The resulting xterm will *not* behave well; rather, as described above, you should access it from the `Mathematical Sciences` folder to get useful keyboard customisations.

### 3.1.2 Running Maple on sysa

Once you have got a login prompt for `sysa` here is how to continue.

1. Enter your username and password; note that you press the `return` key after each entry. An **xterm** window will appear.
2. If you are asked to choose a terminal type, select option 4, and, when prompted for your terminal type, type: `xterm`. You should then have a Unix command prompt in an `xterm` window.
3. The first thing to do is say what software you intend to use, so that some customisations can be made to your login session. Type: `use maple7`. You will need to repeat this step each time you log on.
4. If the previous step worked, it will have resulted in a one-line message; otherwise there will be additional instructions on the screen which you should follow.
5. (**First time only**) If this is going to be your *first* time using Maple, I suggest you create a directory called mx3503. To do this type: `mkdir mx3503`.
6. Change to your mx3503 directory by typing: `cd mx3503`. This is where your Maple files will be kept. Every time you log on to work with Maple for this course you must chnage to this directory before you start.
7. Now you can start the Maple software with the 'xmaple' command. Type: `xmaple &`
8. When you are finished, quit Maple and then log out of X-windows by typing: `exit` in your xterm window.
9. You should now log out of the PC. Refer back to the appropriate section above for instructions.

You really have no use for the command line prompt except to start MAPLE. In fact the prompt will behave rather like the *bash* prompt you may have met in the JAVA course; indeed if you type "`bash`" the behaviour is extremely similar, but in this case you will be running on a native Unix system.

### 3.1.3 Saving your work

You can save your work between sessions by selecting `File -> Save`; in other words, by choosing the "Save option from the "File" menu. Note that there is *no* default file name offered; you are shown the pattern `*.mws`, meaning you are expected to use a name like `myfile.mws` with the given suffix, but you need to alter the `*` as appropriate. You may wish to save space by selecting `Edit -> Remove Output` before saving, since you can always re-run the sheet to regenerate the output.

If you started MAPLE by first changing to your mx3503 directory, you should find that MAPLE saves your worksheets, by default, in that directory.

### 3.1.4 Printing From Maple

When you try to print from Maple, you will be presented with a dialog box. The first time you print, you must set the appropriate 'Print Command' and 'Paper Size', in the print dialog box. These settings will be remembered by Maple for the next time you use it.

**Print Command** In the Print dialog box, you should select 'Print Command' and replace whatever is there with `lpr -Phold` to send your output to one of the other classrooms' laser printers.

**Paper Size** In the 'paper size' section you should select 'A4'.

## 3.2 Using Maple to Check Results

As our first example, we consider again the minimisation problem given in Example 2.17. We show what is almost a transcription of the session with MAPLE; note that MAPLE commands appear after the prompt `>`, while the responses, or answers, appear more or less in the centre of the line or lines below:

```
> with(simplex):
```

The package to do the manipulations required by the simplex algorithm is not available to MAPLE unless you ask for it. We ask for the package with the command `with(simplex);` and the response shows the new commands available. You can probably guess what some of them do. To find out more, type `?simplex;` and a help screen will appear. To get information about the individual commands, you can query them within the package; thus `?simplex,setup` gives information about the setup command. Before going any further, I note that the syntax is important. Any MAPLE command ends with a semicolon ";". without this punctuation the system will wait for further input. A second syntactic point is about *assignment*. The command `x:=2` assigns `x` the value `2`. In contrast the statement `x=2` is a predicate; it has the value true or false depending on the value of `x` before the test, and the value of `x` is not affected by the test. This is the same as Pascal, but different from the FORTRAN convention, in which the `=` on its own is used for assignment. To make the point more strongly, you can check the following conversation with MAPLE.

```
> x=2;
```

$$x = 2$$

```
> x;
```

$$x$$

```
> x:=2;
```

$$x := 2$$

```
> x;
```

$$2$$

Another point to note concerns "unassigning" variables. If you have been manipulating the variable `x`, and wish to re-run some commands, you should unassign `x` before doing so with the command `x := 'x';`. If this is not done, you may find that `x` is contaminated by a previous run.

To return to the simplex algorithm, we specify the constraints and set up the problem:

```
> constraints:=[x1-2*x2 >= -4, 2*x1 + 3*x2 <= 13, x1-x2 <=4];
```

$$constraints := [-4 \leq x1 - 2\,x2,\ 2\,x1 + 3\,x2 \leq 13,\ x1 - x2 \leq 4]$$

We give the constraints in an arbitrary form, as a list, with the individual constraints delimited by commas. All the documentation suggests that constraints can be specified either as a list (with square brackets thus `[a,b,c]`) or as a set thus `{a ,b, c}`. In practice there seems to be a bug in this version of the code, MAPLE Version 5.3 which means that sets are not always handled properly. The difference between the two types is that in a list, the elements are ordered, while in a set, they are not; it seems that some code was relying on the fact that a set of ratios built from a set of constraints was ordered in the same way as the constraint set. You are thus advised *only* to use lists for constraints until the problem is solved. For reference, given lists `L1 = [a,b]` and `L2 = [c,d]`, the list `L = [a,b,c,d]` is made by

```
L1 := [a,b]; L2 := [c,d]; L:=[op(L1), op(L2)];
```

More generally, note that the comma separator needs to be used when creating or manipulating both sets and lists.

If you have a set, say `a:= {x, y, z}`, then the corresponding list is made by

```
convert(a,list);
```

which produces `[x y z]`. The `op` used above selects operands; all of them when we used it to get the elements of the lists `L1` and `L2`; more generally, `op(2, L)` is just `b`.

Given now a list of constraints, it can then be converted to the standard $\leq$ form for subsequent processing.

```
> c:=convert(constraints,stdle);
```

$$c := [-x1 + 2\,x2 \leq 4,\ 2\,x1 + 3\,x2 \leq 13,\ x1 - x2 \leq 4]$$

```
> c:=setup(c,NONNEGATIVE);
```

$$c := [\_SL1 = 4 + x1 - 2\,x2,\ \_SL2 = 13 - 2\,x1 - 3\,x2,\ \_SL3 = 4 - x1 + x2]$$

The problem is now set up, and MAPLE automatically introduces slack variables, which it calls `_SL1, _SL2, _SL3` to convert the inequalities to equalities.

```
> basis(c);
```

$$[\_SL1,\ \_SL2,\ \_SL3]$$

```
> obj:=-4*x1+3*x2;
```

$$obj := -4\,x1 + 3\,x2$$

The basis of the set of constraints is another way of describing the set of basic columns in the first tableau of Table 2.5. Our aim is to minimise the objective function which we call `obj`. Note that, because we are dealing with the 1-phase problem, we *have* a basic feasible solution already; the rather boring one $x_1 = x_2 = 0$.

```
> pivotvar(-obj);
```

$$x1$$

```
> ratio(c,x1);
```

$$[\infty, \frac{13}{2}, 4]$$

Now we set about improving the objective function. The command `pivotvar` only has the concept of a maximising problem, so we try to maximise `-obj`. In any case the idea is clear; to *decrease* the objective function, while keeping $x_1$ and $x_2$ non-negative, we clearly start by working on $x_1$, rather than $x_2$. The `ratio` command looks in the last column of the tableau to see which equation to swap out of the basis. Recalling Theorem 2.16, we want the smallest positive ratio, provided it is non-zero. MAPLE chooses to report negative ratios as $\infty$ so there can be no confusion, and we see that we should swap out the third equation, corresponding to a ratio of 4.

```
> e:=pivoteqn(c,x1);
```

$$e := [\_SL3 = 4 - x1 + x2]$$

```
> c:=pivot(c,x1,e);
```

$$c := [\_SL1 = 8 - \_SL3 - x2,\ \_SL2 = 5 + 2\,\_SL3 - 5\,x2,\ x1 = -\_SL3 + 4 + x2]$$

```
> basis(c);
```

$$[\_SL1,\ \_SL2,\ x1]$$

We have now set `e` to the equation to be swapped out (the command `pivoteqn` has used the command `ratio` itself), and done the pivot, to put $x_1$ in the basis; we see that `_SL3` has come out of the basis. The constraints are then re-arranged so that the basis variables appear on the left hand side of the constraint equations; this is the new value of the constraint set `c`. Initially, we could set $x_1$ and $x_2$ as we wished, and the slack variables were determined by these choices. Now our "free" variables are $x_2$ and `_SL3`, and the others are then determined by the constraints.

```
> obj:=subs(c,obj);
```

$$obj := 4\,\_SL3 - 16 - x2$$

Before going on, we express the objective in terms of the new "free" variables, and then repeat the process. The command `pivotvar` reports Fail if there is no proper sign for improvement. In our case, we can continue as follows:

```
> pivotvar(-obj);
```

$$x2$$

```
> ratio(c,x2);
```

$$[8,\ 1,\ \infty]$$

```
> e:=pivoteqn(c,x2);
```

$$e := [\_SL2 = 5 + 2\,\_SL3 - 5\,x2]$$

```
> c:=pivot(c,x2,e);
```

$$c := [\_SL1 = 7 - \frac{7}{5}\,\_SL3 + \frac{1}{5}\,\_SL2,\ x2 = -\frac{1}{5}\,\_SL2 + 1 + \frac{2}{5}\,\_SL3,\ x1 = -\frac{3}{5}\,\_SL3 + 5 - \frac{1}{5}\,\_SL2$$
]

```
> basis(c);
```

$$[\_SL1,\ x2,\ x1]$$

```
> obj:=subs(c,obj);
```

$$obj := \frac{18}{5}\,\_SL3 - 17 + \frac{1}{5}\,\_SL2$$

In fact the algorithm has now terminated; the objective function has no proper sign for improvement. To find the "answers", we substitute zero for each of the free (or non-basis) variables:

```
> free:=[_SL2=0,_SL3=0];
```

$$free := [\_SL2 = 0,\ \_SL3 = 0]$$

```
> subs(free,obj);
```

$$-17$$

```
> subs(free,c);
```

$$[\_SL1 = 7,\ x2 = 1,\ x1 = 5]$$

And of course we get the same solution as before.

*3.1. Example.* Give a MAPLE solution for Example 2.20.

*Solution* We give the dialogue with MAPLE without further comment.

```
> with(simplex):
> c:=[3*x1 + 4*x2 <=12, 7*x1+2*x2 <=14];
```

$$c := [3\,x1 + 4\,x2 \le 12,\ 7\,x1 + 2\,x2 \le 14]$$

```
> c:=setup(c,NONNEGATIVE);
```

$$c := [\_SL1 = 12 - 3\,x1 - 4\,x2,\ \_SL2 = 14 - 7\,x1 - 2\,x2]$$

```
> obj:=4*x1+3*x2;
```

$$obj := 4\,x1 + 3\,x2$$

```
> pivotvar(obj);
```

$$x1$$

```
> ratio(c,x1);
```

$$[4,\ 2]$$

```
> c:=pivot(c,x1,pivoteqn(c,x1));
```

$$c := [\_SL1 = 6 + \frac{3}{7}\,\_SL2 - \frac{22}{7}\,x2,\ x1 = -\frac{1}{7}\,\_SL2 + 2 - \frac{2}{7}\,x2]$$

```
> obj:=subs(c,obj);
```

$$obj := -\frac{4}{7}\,\_SL2 + 8 + \frac{13}{7}\,x2$$

```
> pivotvar(obj);
```

$$x2$$

```
> ratio(c,x2);
```

$$[\frac{21}{11},\ 7]$$

```
> c:=pivot(c,x2,pivoteqn(c,x2));
```

$$c := [x2 = -\frac{7}{22}\,\_SL1 + \frac{21}{11} + \frac{3}{22}\,\_SL2,\ x1 = -\frac{2}{11}\,\_SL2 + \frac{16}{11} + \frac{1}{11}\,\_SL1]$$

```
> obj:=subs(c,obj);
```

$$obj := -\frac{7}{22}\,\_SL2 + \frac{127}{11} - \frac{13}{22}\,\_SL1$$

```
> free:=[_SL1=0,_SL2=0];
```

$$free := [\_SL1 = 0,\ \_SL2 = 0]$$

```
> subs(free,obj);
```

$$\frac{127}{11}$$

```
> subs(free,c);
```

$$[x2 = \frac{21}{11},\ x1 = \frac{16}{11}]$$

It is not *necessary* to work through the example by hand as we have just done: if you are *just* interested in the answer, then the built in commands of `maximize` and `minimize` give the answer almost immediately. Here is a minimal version of the first example:

```
> with(simplex):
> c:=[x1-2*x2 >= -4, 2*x1 + 3*x2 <=13, x1-x2 <=4];
```

$$c := [-4 \le x1 - 2\,x2,\ 2\,x1 + 3\,x2 \le 13,\ x1 - x2 \le 4]$$

```
> c:=setup(c);
```

$$c := [\_SL1 = 4 + x1 - 2\,x2,\ \_SL2 = 13 - 2\,x1 - 3\,x2,\ \_SL3 = 4 - x1 + x2]$$

```
> obj:=-4*x1+3*x2;
```

$$obj := -4\,x1 + 3\,x2$$

```
> vars:=minimize(obj,c,NONNEGATIVE);
```

$$vars := \{\_SL3 = 0,\ \_SL2 = 0,\ \_SL1 = 7,\ x2 = 1,\ x1 = 5\}$$

```
> subs(vars,obj);
```

$$-17$$

It is perhaps appropriate to note two points here. Even though we work with lists, `minimize` returns a set. And minimize is written exactly as given here; the function `minimise` is undefined!

## 3.3 Pivoting Using Maple

An alternative way of using MAPLE is simply to let it do the pivoting you would otherwise do by hand. Here is what happens if we do the first example using the tableau method we have described above, making the same decisions about which rows to swap in and out, and simply using MAPLE to do the pivoting. As you will see, the main problem is in constructing the tableau in the first place!

```
> with(linalg):
```

We don't need the `simplex` package; the more general purpose, but much larger linear algebra package, declared as shown, has what we need; in fact it overrides some of the definitions in the simplex package. The linear algebra package contains all the above routines

in it. Recall you can get help on any routine with the command eg `? linalg,pivot`. You can probably guess enough to ask sensible questions of the help system; you may find you can get enough information in this way to be able to use MAPLE for work in other subjects.

We first set up the tableau.

```
> A:=concat(matrix(3,2,[-1,2,2,3,1,-1]),diag(1,1,1));
```

$$A := \begin{bmatrix} -1 & 2 & 1 & 0 & 0 \\ 2 & 3 & 0 & 1 & 0 \\ 1 & -1 & 0 & 0 & 1 \end{bmatrix}$$

```
> B:=concat(A,concat(vector([0,0,0]),vector([4,13,4])));
```

$$B := \begin{bmatrix} -1 & 2 & 1 & 0 & 0 & 0 & 4 \\ 2 & 3 & 0 & 1 & 0 & 0 & 13 \\ 1 & -1 & 0 & 0 & 1 & 0 & 4 \end{bmatrix}$$

```
> obj:=vector([4,-3,0,0,0,1,0]);
```

$$obj := [4, -3, 0, 0, 0, 1, 0]$$

```
> T1:=stackmatrix(A,obj);
```

$$T1 := \begin{bmatrix} -1 & 2 & 1 & 0 & 0 & 0 & 4 \\ 2 & 3 & 0 & 1 & 0 & 0 & 13 \\ 1 & -1 & 0 & 0 & 1 & 0 & 4 \\ 4 & -3 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

We have now made the first of the tableaus in Table 2.5. The calculation continues exactly as there; all we get MAPLE to do is perform the pivoting operations.

```
> T2:=pivot(T1,3,1);
```

$$T2 := \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 8 \\ 0 & 5 & 0 & 1 & -2 & 0 & 5 \\ 1 & -1 & 0 & 0 & 1 & 0 & 4 \\ 0 & 1 & 0 & 0 & -4 & 1 & -16 \end{bmatrix}$$

```
> T3:=pivot(T2,2,2);
```

$$T3 := \begin{bmatrix} 0 & 0 & 1 & \frac{-1}{5} & \frac{7}{5} & 0 & 7 \\ 0 & 5 & 0 & 1 & -2 & 0 & 5 \\ 1 & 0 & 0 & \frac{1}{5} & \frac{3}{5} & 0 & 5 \\ 0 & 0 & 0 & \frac{-1}{5} & \frac{-18}{5} & 1 & -17 \end{bmatrix}$$

And of course we end up with the same tableau — except that MAPLE chooses not to normalise a row before using it to pivot.

## 3.4 Cycling in Example 2.26

In section 2.8 we asserted that the algorithm could cycle, given a particular choice of basis sequence. We now verify this using MAPLE.

```
> with(linalg):
```

```
> M:=array([[1/4,-8,-1,9],[1/2,-12,-1/2,3],[0,0,1,0],[-1/4,20,-1/2,6]])
> ;
```

$$M := \begin{bmatrix} \frac{1}{4} & -8 & -1 & 9 \\ \frac{1}{2} & -12 & \frac{-1}{2} & 3 \\ 0 & 0 & 1 & 0 \\ \frac{-1}{4} & 20 & \frac{-1}{2} & 6 \end{bmatrix}$$

```
> MI:=concat(M,diag(1,1,1,1));
```

$$MI := \begin{bmatrix} \frac{1}{4} & -8 & -1 & 9 & 1 & 0 & 0 & 0 \\ \frac{1}{2} & -12 & \frac{-1}{2} & 3 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ \frac{-1}{4} & 20 & \frac{-1}{2} & 6 & 0 & 0 & 0 & 1 \end{bmatrix}$$

```
> b:=vector([0,0,1,0]);
```

$$b := [0,\ 0,\ 1,\ 0]$$

```
> A:=concat(MI,b);B:=copy(A):
```

$$A := \begin{bmatrix} \frac{1}{4} & -8 & -1 & 9 & 1 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & -12 & \frac{-1}{2} & 3 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ \frac{-1}{4} & 20 & \frac{-1}{2} & 6 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

You might have expected to store a copy of the matrix `A` in `B` by using the command `B:=A;`, but this would not have done what you wished. It would have made `B` into a pointer which always gave you the *current* value of `A`, even when it had altered. Since I am trying to show that after a number of changes `A` comes back to where it started, I need to copy the original *value* of `A` into `B`.

```
> A:=pivot(A,1,1);
```

$$A := \begin{bmatrix} \frac{1}{4} & -8 & -1 & 9 & 1 & 0 & 0 & 0 & 0 \\ 0 & 4 & \frac{3}{2} & -15 & -2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 12 & \frac{-3}{2} & 15 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

```
> A:=pivot(A,2,2);
```

$$A := \begin{bmatrix} \frac{1}{4} & 0 & 2 & -21 & -3 & 2 & 0 & 0 & 0 \\ 0 & 4 & \frac{3}{2} & -15 & -2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & -6 & 60 & 7 & -3 & 0 & 1 & 0 \end{bmatrix}$$

```
> A:=pivot(A,1,3);
```

$$A := \begin{bmatrix} \frac{1}{4} & 0 & 2 & -21 & -3 & 2 & 0 & 0 & 0 \\ \frac{-3}{16} & 4 & 0 & \frac{3}{4} & \frac{1}{4} & \frac{-1}{2} & 0 & 0 & 0 \\ \frac{-1}{8} & 0 & 0 & \frac{21}{2} & \frac{3}{2} & -1 & 1 & 0 & 1 \\ \frac{3}{4} & 0 & 0 & -3 & -2 & 3 & 0 & 1 & 0 \end{bmatrix}$$

```
> A:=pivot(A,2,4);
```

$$A := \begin{bmatrix} -5 & 112 & 2 & 0 & 4 & -12 & 0 & 0 & 0 \\ \frac{-3}{16} & 4 & 0 & \frac{3}{4} & \frac{1}{4} & \frac{-1}{2} & 0 & 0 & 0 \\ \frac{5}{2} & -56 & 0 & 0 & -2 & 6 & 1 & 0 & 1 \\ 0 & 16 & 0 & 0 & -1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

```
> A:=pivot(A,1,5);
```

$$A := \begin{bmatrix} -5 & 112 & 2 & 0 & 4 & -12 & 0 & 0 & 0 \\ \frac{1}{8} & -3 & \frac{-1}{8} & \frac{3}{4} & 0 & \frac{1}{4} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ \frac{-5}{4} & 44 & \frac{1}{2} & 0 & 0 & -2 & 0 & 1 & 0 \end{bmatrix}$$

```
> A:=pivot(A,2,6);
```

$$A := \begin{bmatrix} 1 & -32 & -4 & 36 & 4 & 0 & 0 & 0 & 0 \\ \frac{1}{8} & -3 & \frac{-1}{8} & \frac{3}{4} & 0 & \frac{1}{4} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ \frac{-1}{4} & 20 & \frac{-1}{2} & 6 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

```
> A:=mulrow(A,1,1/4):A:=mulrow(A,2,4);
```

$$A := \begin{bmatrix} \frac{1}{4} & -8 & -1 & 9 & 1 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & -12 & \frac{-1}{2} & 3 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ \frac{-1}{4} & 20 & \frac{-1}{2} & 6 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

```
> matadd(A,B,1,-1);
```

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Thus we have completed a cycle. Note that `A` and `B` were assigned the same value, so this is a check that we have returned to where we started. If instead I had done `B:=A`, it would have shown nothing, because the value of `B` would have been the "current" value of `A`.

Cycling occurred when we made a particular choice of pivot. Starting in a different way leads to convergence.

```
> A:=pivot(A,2,1);
```

$$A := \begin{bmatrix} 0 & -2 & \frac{-3}{4} & \frac{15}{2} & 1 & \frac{-1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & -12 & \frac{-1}{2} & 3 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 14 & \frac{-3}{4} & \frac{15}{2} & 0 & \frac{1}{2} & 0 & 1 & 0 \end{bmatrix}$$

```
> A:=pivot(A,3,3);
```

$$A := \begin{bmatrix} 0 & -2 & 0 & \frac{15}{2} & 1 & \frac{-1}{2} & \frac{3}{4} & 0 & \frac{3}{4} \\ \frac{1}{2} & -12 & 0 & 3 & 0 & 1 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 14 & 0 & \frac{15}{2} & 0 & \frac{1}{2} & \frac{3}{4} & 1 & \frac{3}{4} \end{bmatrix}$$

```
> A:=mulrow(A,2,2);
```

$$A := \begin{bmatrix} 0 & -2 & 0 & \frac{15}{2} & 1 & \frac{-1}{2} & \frac{3}{4} & 0 & \frac{3}{4} \\ 1 & -24 & 0 & 6 & 0 & 2 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 14 & 0 & \frac{15}{2} & 0 & \frac{1}{2} & \frac{3}{4} & 1 & \frac{3}{4} \end{bmatrix}$$

Having pivoted about $(3, 3)$, we see there was no proper sign for improvement; thus we can read off the maximum value: $3/4$ as claimed.

## 3.5 An Extended Exercise

You are now in a position to tackle completely the request from the board of MMM described originally in 1.3. You are reminded that this is to take the form of the report that you, as their consultant, deliver to the board, recommending their course of action with respect to the three new alloys. I'm sure that writing such a report will impose significantly on your already busy schedule; however this type of tight timing also occurs in practice, so this is not an inappropriate simulation. Since the point of this exercise is mainly one of

presentation and the *use* of MAPLE, rather than in solving the problem, I've made available one MAPLE "solution" to the problem on the web

# Chapter 4

# The Two-Phase Simplex Algorithm

## 4.1 Introduction

In the one-phase simplex algorithm it is assumed that an initial basic feasible solution for the related standard form linear programming problem is either given or obvious.

Suppose, for instance, that the original problem is: maximise $x_{n+1}$ subject to

$$\mathbf{Ax} \leq \mathbf{b} \quad \text{and } x_1, x_2, \ldots, x_n \geq 0$$

where $\mathbf{b} \geq 0$. This is the particular case when the constraints can all be written as "less than or equals" and the right-hand sides turn out to be greater than or equal to 0. The related standard form is obtained by adding $m$ slack variables and $\mathbf{x} = (0, 0, \ldots, 0, b_1, \ldots, b_m; 0)$ is obviously an initial basic feasible solution. In general it is necessary to have an algorithm, called the **second phase**, to find a basic feasible solution, after which the "first phase" already discussed is used to complete the solution of the problem.

| | $\mathbf{a}_1$ | $\mathbf{a}_2$ | ... | $\mathbf{a}_s$ | ... | $\mathbf{a}_n$ | $\mathbf{e}$ | $\mathbf{b}$ | |
|---|---|---|---|---|---|---|---|---|---|
| $\mathbf{a}_{h_1}$ | | | | $y_{1s}$ | | | 0 | $u_1$ | $(\geq 0)$ |
| $\vdots$ | | | | $\vdots$ | | | $\vdots$ | $\vdots$ | |
| $\mathbf{a}_{h_i}$ | | | | $y_{is}$ | | | 0 | $u_i$ | $(\geq 0)$ |
| $\vdots$ | | | | $\vdots$ | | | $\vdots$ | $\vdots$ | |
| $\mathbf{a}_{h_q}$ | | | | $y_{qs}$ | | | 0 | $u_q$ | $(\geq 0)$ |
| $\mathbf{a}_{h_{q+1}}$ | | | | $y_{q+1,s}$ | | | 0 | $u_{q+1}$ | $(< 0)$ |
| $\vdots$ | | | | $\vdots$ | | | $\vdots$ | $\vdots$ | |
| $\mathbf{a}_{h_m}$ | | | | $y_{ms}$ | | | 0 | $u_m$ | $(< 0)$ |
| $\mathbf{e}$ | $v_1$ | $v_2$ | ... | $v_s$ | ... | $v_n$ | 1 | $z$ | |

Table 4.1: A two phase tableau.

**4.1. Definition.** A **two-phase tableau** for $A\mathbf{x} = \mathbf{b}$ is a tableau with $\mathbf{e}$ as the last member of the basis and with the rows of the tableau arranged so that all the negative $u_i$, if any, are below all the non-negative $u_i$ in the last column.

Table 4.1 shows a two-phase tableau. The rows 1 to $q$ in which $u_i \geq 0$ are called **good** rows and the rows $q+1$ to $m$ in which $u_i < 0$ are **bad** rows. The associated basic solution is a basic *feasible* solution if and only if $q = m$.

## 4.2 The Second Phase Described

As with the one-phase we use a flow chart to summarise the behaviour. We assume in Fig. 4.1 that we have a genuine two-phase problem. The chart then shows how to *reduce* the problem to the one phase situation, at which point we can use the flowchart of Fig. 2.1.

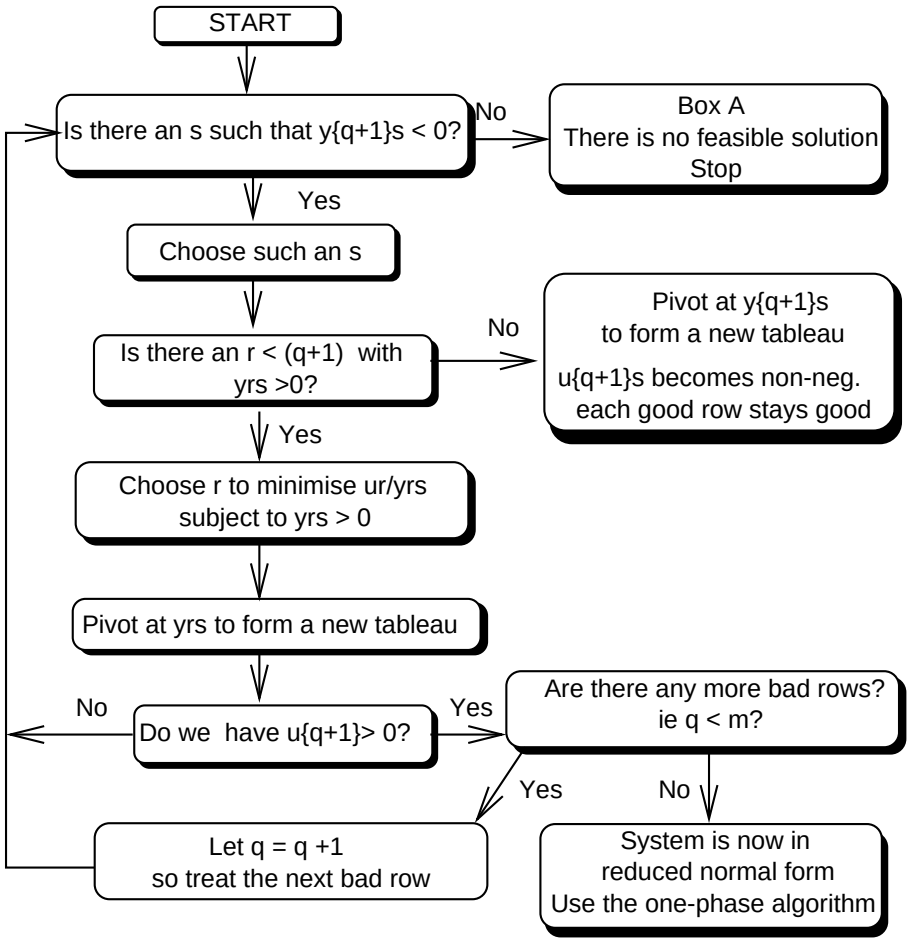


Figure 4.1: Flow chart for two-phase simplex algorithm.

If $q = m$, proceed immediately to the first-phase. If not, treat the first bad row as the objective row in a maximising problem with the aim of increasing $u_{q+1}$ until it becomes greater than or equal to 0. Thus, find a $y_{q+1,s} < 0$, if possible; if it is not possible, we shall see that there cannot be any feasible solutions. Then, if possible, choose from the good rows a suitable $r$ (as in the one-phase method) and pivot about $y_{rs}$, thereby increasing the value of $u_{q+1}$. If there is no such $r$ then pivot about $y_{q+1,s}$ itself. In this way, the aim is to change each bad row in turn into a good row. Fig. 4.1 shows a flow-chart that gives more details.

We will first consider an example in which all the constraints are inequalities. The general procedure is to make all the constraints into "less than or equals" by multiplying by $-1$ if necessary. Then the constraints are rearranged so that those in which the right hand side is negative come below the others. As in the one-phase algorithm, introduce slack variables and add the objective row.

*4.2. Example.* Maximise $5x_1 - 3x_2 + 10$ subject to

$$\begin{aligned} 2x_1 - 3x_2 &\geq 6, \\ x_1 + 4x_2 &\leq 15, \\ 2x_1 + 5x_2 &\geq -4, \\ x_1, x_2 &\geq 0. \end{aligned}$$

*Solution* A related standard form for this problem is:

Maximise $x_6$ subject to

$$\begin{aligned} x_1 + 4x_2 + x_3 &= 15, \\ -2x_1 - 5x_2 + x_4 &= 4, \\ -2x_1 + 3x_2 + x_5 &= -6, \\ -5x_1 + 3x_2 + x_6 &= 10, \\ x_1, \ldots, x_5 &\geq 0. \end{aligned}$$

Table 4.2 shows the tableaus obtained by following the two-phase algorithm. The initial tableau $T_1$ of course has an associated basic solution $(0, 0, 15, 4, -6; 10)$, but notice that this is not feasible. Pivoting about $y_{11}$ $(= 1)$ gives tableau $T_2$ with basic *feasible* solution $(15, 0, 0, 0, 34, 24; 85)$ which is seen to be optimal. So an optimal solution of the original problem is $x_1 = 15, x_2 = 0$ with optimal value 85.

| | | $\mathbf{a}_1$ | $\mathbf{a}_2$ | $\mathbf{a}_3$ | $\mathbf{a}_4$ | $\mathbf{a}_5$ | $\mathbf{e}$ | $\mathbf{b}$ |
|---|---|---|---|---|---|---|---|---|
| $T_1$ | $\mathbf{a}_3$ | $1^*$ | 4 | 1 | 0 | 0 | 0 | 15 |
| | $\mathbf{a}_4$ | $-2$ | $-5$ | 0 | 1 | 0 | 0 | 4 |
| | $\mathbf{a}_5$ | $-2$ | 3 | 0 | 0 | 1 | 0 | $-6$ |
| | $\mathbf{e}$ | $-5$ | 3 | 0 | 0 | 0 | 1 | 10 |
| $T_2$ | $\mathbf{a}_1$ | 1 | 4 | 1 | 0 | 0 | 0 | 15 |
| | $\mathbf{a}_4$ | 0 | 3 | 2 | 1 | 0 | 0 | 34 |
| | $\mathbf{a}_5$ | 0 | 11 | 2 | 0 | 1 | 0 | 24 |
| | $\mathbf{e}$ | 0 | 23 | 5 | 0 | 0 | 1 | 85 |

Table 4.2: Two tableaus for Example 4.2.

*4.3. Example.* Consider the same problem as Example 4.2 except that $y_{11} = 0$ instead of 1.

*Solution* In these circumstances, one uses Box B and pivots about an entry in the bad row itself — pivot about $y_{31}(= -2)$. Table 4.3 shows the result. Tableau $T_2$ has basic feasible solution $(3, 0, 15, 10, 0; 25)$ and this time one sees that there is *no* optimal solution. This

is because $v_5(=-\frac{5}{2})$ has the proper sign for improvement, but none of the entries in that column is positive.

Note also that $v_2(=-\frac{9}{2})$ also has the proper sign for improvement, and we could indeed pivot about the corresponding entry 4 in row 1; but our Theorem 2.23 tells us that there is no optimal solution anyway; a further pivot merely postpones the realization of this fact.

Finally note that in the tableau $T_1$, column 1 has all its entries $\leq 0$. We can't apply our results of Theorem 2.23 directly, since we don't yet have a feasible solution, and the result was only proved assuming we started from a tableau *with* a basic feasible solution.

| | | $\mathbf{a}_1$ | $\mathbf{a}_2$ | $\mathbf{a}_3$ | $\mathbf{a}_4$ | $\mathbf{a}_5$ | $\mathbf{e}$ | $\mathbf{b}$ |
|---|---|---|---|---|---|---|---|---|
| $T_1$ | $\mathbf{a}_3$ | 0 | 4 | 1 | 0 | 0 | 0 | 15 |
| | $\mathbf{a}_4$ | $-2$ | $-5$ | 0 | 1 | 0 | 0 | 4 |
| | $\mathbf{a}_5$ | $-2^*$ | 3 | 0 | 0 | 1 | 0 | $-6$ |
| | $\mathbf{e}$ | $-5$ | 3 | 0 | 0 | 0 | 1 | 10 |
| $T_2$ | $\mathbf{a}_3$ | 0 | 4 | 1 | 0 | 0 | 0 | 15 |
| | $\mathbf{a}_4$ | 0 | $-8$ | 0 | 1 | $-1$ | 0 | 10 |
| | $\mathbf{a}_1$ | 1 | $-3/2$ | 0 | 0 | $-1/2$ | 0 | 3 |
| | $\mathbf{e}$ | 0 | $-9/2$ | 0 | 0 | $-5/2$ | 1 | 25 |

Table 4.3: Example 4.3: pivoting about a bad row.

*4.4. Example.* Consider Example 4.2 again, but with a different change; with $b_1 = 2$ instead of 15.

*Solution* Pivoting about $y_{11}(=1)$ in tableau $T_1$ of Table 4.4 gives tableau $T_2$ which still has a bad row. Since now none of the entries in this bad row is negative, the value $u_{q+1}(=-2)$ cannot be increased. We therefore reach Box A and the problem has no feasible solutions.

| | | $\mathbf{a}_1$ | $\mathbf{a}_2$ | $\mathbf{a}_3$ | $\mathbf{a}_4$ | $\mathbf{a}_5$ | $\mathbf{e}$ | $\mathbf{b}$ |
|---|---|---|---|---|---|---|---|---|
| $T_1$ | $\mathbf{a}_3$ | $1^*$ | 4 | 1 | 0 | 0 | 0 | 2 |
| | $\mathbf{a}_4$ | $-2$ | $-5$ | 0 | 1 | 0 | 0 | 4 |
| | $\mathbf{a}_5$ | $-2$ | 3 | 0 | 0 | 1 | 0 | $-6$ |
| | $\mathbf{e}$ | $-5$ | 3 | 0 | 0 | 0 | 1 | 10 |
| $T_2$ | $\mathbf{a}_1$ | 1 | 4 | 1 | 0 | 0 | 0 | 2 |
| | $\mathbf{a}_4$ | 0 | 3 | 2 | 1 | 0 | 0 | 8 |
| | $\mathbf{a}_5$ | 0 | 11 | 2 | 0 | 1 | 0 | $-2$ |
| | $\mathbf{e}$ | 0 | 23 | 5 | 0 | 0 | 1 | 20 |

Table 4.4: Example 4.4 with no feasible solutions.

## 4.3 Artificial Variables

When any of the constraints is an equation, there are two possible approaches. In one we replace the equation by a pair of inequalities as discussed in Section 1.1. An alternative way, sometimes significantly easier to manage by hand is to introduce a corresponding **artificial variable**.

In the following example, for the equation $x_1 + 4x_2 = 15$, an artificial variable $x_3$ is introduced to give $x_1 + 4x_2 + x_3 = 15$. Then a basic feasible solution of the new problem corresponds to a basic feasible solution of the original problem only if $x_3 = 0$. The new problem is strictly speaking not a related standard form of the original. The procedure is to make a change of basis in order to replace $\mathbf{a}_3$ and thus ensure that $x_3 = 0$. The column $\mathbf{a}_3$ can then be ignored and the two-phase algorithm followed.

The reason for the introduction of an artificial variable is that otherwise one doesn't have a genuine tableau with which to start: the matrix obtained is not in canonical form.

*4.5. Example.* Maximise $5x_1 - 3x_2 + 10$ subject to

$$\begin{aligned} 2x_1 - 3x_2 &\geq 6, \\ x_1 + 4x_2 &= 15, \\ 2x_1 + 5x_2 &\geq -4, \\ x_1, x_2 &\geq 0. \end{aligned}$$

*Solution* We obtain the following Linear Programming Problem in standard form: maximise $x_6$ subject to

$$\begin{aligned} x_1 + 4x_2 + x_3 &= 15, \\ -2x_1 - 5x_2 + x_4 &= 4, \\ -2x_1 + 3x_2 + x_5 &= -6, \\ -5x_1 + 3x_2 + x_6 &= 10, \\ x_1, ..., x_5 &\geq 0, \end{aligned}$$

where the variable $x_3$ is artificial. Table 4.5 shows that pivoting about $y_{11}$ $(= 1)$ in the first row removes $\mathbf{a}_3$ from the basis. It happens that as a result the value of $u_{q+1}(= -6)$ is increased and $T_2$ has no bad rows. The procedure is now to ignore column $\mathbf{a}_3$ and continue.

In this case we have already arrived at an optimal solution $(15, 0, 0, 34, 24; 85)$ which has $x_3 = 0$ as required. So $x_1 = 15$, $x_2 = 0$ is an optimal solution of the original problem with optimal value 85.

### 4.3.1 The $M$ method

There is an interesting re-arrangement of this calculation known as the **big $M$ method**. In this form we deal with the artificial variable by modifying the objective function. Thus in Example 4.5, a maximising problem in which $x_3$ is the artificial variable we work with the objective function

$$x_6 = 5x_1 - 3x_2 + 10 - Mx_3,$$

| | | $\mathbf{a}_1$ | $\mathbf{a}_2$ | $\mathbf{a}_3$ | $\mathbf{a}_4$ | $\mathbf{a}_5$ | $\mathbf{e}$ | $\mathbf{b}$ |
|---|---|---|---|---|---|---|---|---|
| $T_1$ | $\mathbf{a}_3$ | $1^*$ | 4 | 1 | 0 | 0 | 0 | 15 |
| | $\mathbf{a}_4$ | $-2$ | $-5$ | 0 | 1 | 0 | 0 | 4 |
| | $\mathbf{a}_5$ | $-2$ | 3 | 0 | 0 | 1 | 0 | $-6$ |
| | $\mathbf{e}$ | $-5$ | 3 | 0 | 0 | 0 | 1 | 10 |
| $T_2$ | $\mathbf{a}_1$ | 1 | 4 | 1 | 0 | 0 | 0 | 15 |
| | $\mathbf{a}_4$ | 0 | 3 | 2 | 1 | 0 | 0 | 34 |
| | $\mathbf{a}_5$ | 0 | 11 | 2 | 0 | 1 | 0 | 24 |
| | $\mathbf{e}$ | 0 | 23 | 5 | 0 | 0 | 1 | 85 |

Table 4.5: Example 4.5: artificial variables

where $M$ is a large positive constant. There is no need to specify in advance how large it is; instead insist that it is larger than any given competitor, so that for example $M - 7$ or $M - 63$ are each guaranteed to be positive. Note that $x_3 \geq 0$, and that we have constructed our auxiliary objective function in such a way that $x_6$ is reduced when $x_3 > 0$. Since $M$ is so large, a maximum of $x_6$ is bound to occur when $x_3 = 0$, unless the original problem does not have a feasible solution. We thus set the problem up as follows:

```
> with(linalg):
> A:=matrix(4,7,[1,4,1,0,0,0,15,-2,-5,0,1,0,0,4,-2,3,0,0,1,0,-6,-5,3,M,
> 0,0,1,10]);
```

$$A := \begin{bmatrix} 1 & 4 & 1 & 0 & 0 & 0 & 15 \\ -2 & -5 & 0 & 1 & 0 & 0 & 4 \\ -2 & 3 & 0 & 0 & 1 & 0 & -6 \\ -5 & 3 & M & 0 & 0 & 1 & 10 \end{bmatrix}$$

Note that this does *not* give a tableau, but we soon restore one by doing the obvious pivot.

```
> B:=pivot(A,1,3);
```

$$B := \begin{bmatrix} 1 & 4 & 1 & 0 & 0 & 0 & 15 \\ -2 & -5 & 0 & 1 & 0 & 0 & 4 \\ -2 & 3 & 0 & 0 & 1 & 0 & -6 \\ -M-5 & -4M+3 & 0 & 0 & 0 & 1 & -15M+10 \end{bmatrix}$$

At this point we have a "normal" problem to solve; the only unusual feature is in looking for a proper sign for improvement, when we have to remember that $M > 0$ is large when determining the signs in the objective row. Thus in this case both columns 1 and 2 are negative; we choose to pivot in column 1.

```
> C:=pivot(B,1,1);
```

$$C := \begin{bmatrix} 1 & 4 & 1 & 0 & 0 & 0 & 15 \\ 0 & 3 & 2 & 1 & 0 & 0 & 34 \\ 0 & 11 & 2 & 0 & 1 & 0 & 24 \\ 0 & 23 & M+5 & 0 & 0 & 1 & 85 \end{bmatrix}$$

This is our final tableau; since $M > 0$ is large, certainly $M > 5$.

### 4.3.2 Another Example

*4.6. Example.* Minimise $2x_1 - 3x_2 + x_3$ subject to

$$\begin{aligned} 3x_1 - 2x_2 + x_3 &\leq 5, \\ x_1 + 3x_2 - 4x_3 &\leq 9, \\ x_2 + 5x_3 &\geq 1, \\ x_1 + x_2 + x_3 &= 6, \\ x_1, x_2, x_3 &\geq 0. \end{aligned}$$

| | | $\mathbf{a}_1$ | $\mathbf{a}_2$ | $\mathbf{a}_3$ | $\mathbf{a}_4$ | $\mathbf{a}_5$ | $\mathbf{a}_6$ | $\mathbf{a}_7$ | $\mathbf{e}$ | $\mathbf{b}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| | $\mathbf{a}_4$ | 3 | −2 | 1 | 1 | 0 | 0 | 0 | 0 | 5 |
| | $\mathbf{a}_5$ | 1 | 3 | −4 | 0 | 1 | 0 | 0 | 0 | 9 |
| $T_1$ | $\mathbf{a}_6$ | 1 | 1 | 1* | 0 | 0 | 1 | 0 | 0 | 6 |
| | $\mathbf{a}_7$ | 0 | −1 | −5 | 0 | 0 | 0 | 1 | 0 | −1 |
| | $\mathbf{e}$ | −2 | 3 | −1 | 0 | 0 | 0 | 0 | 1 | 0 |
| | $\mathbf{a}_4$ | 2 | −3 | 0 | 1 | 0 | −1 | 0 | 0 | −1 |
| | $\mathbf{a}_5$ | 5 | 7 | 0 | 0 | 1 | 4 | 0 | 0 | 33 |
| $T_2$ | $\mathbf{a}_3$ | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 6 |
| | $\mathbf{a}_7$ | 5 | 4 | 0 | 0 | 0 | 5 | 1 | 0 | 29 |
| | $\mathbf{e}$ | −1 | 4 | 0 | 0 | 0 | 1 | 0 | 1 | 6 |
| | $\mathbf{a}_5$ | 5 | 7* | 0 | 0 | 1 | | 0 | 0 | 33 |
| | $\mathbf{a}_3$ | 1 | 1 | 1 | 0 | 0 | | 0 | 0 | 6 |
| $T_3$ | $\mathbf{a}_7$ | 5 | 4 | 0 | 0 | 0 | | 1 | 0 | 29 |
| | $\mathbf{a}_4$ | 2 | −3 | 0 | 1 | 0 | | 0 | 0 | −1 |
| | $\mathbf{e}$ | −1 | 4 | 0 | 0 | 0 | | 0 | 1 | 6 |
| | $\mathbf{a}_2$ | 5/7 | 1 | 0 | 0 | 1/7 | | 0 | 0 | 33/7 |
| | $\mathbf{a}_3$ | 2/7 | 0 | 1 | 0 | −1/7 | | 0 | 0 | 9/7 |
| $T_4$ | $\mathbf{a}_7$ | 15/7 | 0 | 0 | 0 | −4/7 | | 1 | 0 | 71/7 |
| | $\mathbf{a}_4$ | 29/7 | 0 | 0 | 1 | 3/7 | | 0 | 0 | 92/7 |
| | $\mathbf{e}$ | −27/7 | 0 | 0 | 0 | −4/7 | | 0 | 1 | −90/7 |

Table 4.6: Four tableaus for Example 4.6.

*Solution* We obtain the linear programming problem: minimise $x_8$ subject to

$$\begin{aligned} 3x_1 - 2x_2 + x_3 + x_4 &= 5, \\ x_1 + 3x_2 - 4x_3 + x_5 &= 9, \\ x_1 + x_2 + x_3 + x_6 &= 6, \\ -x_2 - 5x_3 + x_7 &= -1, \\ -2x_1 + 3x_2 - x_3 + x_8 &= 0, \\ x_1, \ldots, x_7 &\geq 0, \end{aligned}$$

where $x_6$ is an artificial variable. In tableau $T_1$ of Table 4.6, pivoting about $y_{33}$ $(= 1)$ removes $\mathbf{a}_6$ from the basis. The rows of tableau $T_2$ are then rearranged to give tableau $T_3$ so that the bad row is below the others, and column $\mathbf{a}_6$ is ignored from here on. Pivoting in $T_3$ about $y_{12}$ $(= 7)$ gives tableau $T_4$ which has the basic feasible solution $(0, 33/7, 9/7, 92/7, 0, 0, 71/7, -90/7)$. This has $x_6 = 0$ and is an optimal solution, so $x_1 = 0$, $x_2 = 33/7$, $x_3 = 9/7$ is an optimal solution of the original problem with optimal value $-90/7$.

### 4.3.3 Minimising with the Big $M$ Method

We can use the big $M$ method on this example. Since we are seeking a *minimum* this time, we have as objective function

$$x_8 = 2x_1 - 3x_2 + x_3 + Mx_6.$$

Again we will only obtain a feasible solution of the original problem if in the new problem, $x_6 = 0$. The calculations are straightforward but instructive.

```
> A:=matrix(5,9,[3,-2,1,1,0,0,0,0,5,1,3,-4,0,1,0,0,0,9,1,1,1,0,0,1,0,0,
> 6,0,-1,-5,0,0,0,1,0,-1,-2,3,-1,0,0,-M,0,1,0]);
```

$$A := \begin{bmatrix} 3 & -2 & 1 & 1 & 0 & 0 & 0 & 0 & 5 \\ 1 & 3 & -4 & 0 & 1 & 0 & 0 & 0 & 9 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 6 \\ 0 & -1 & -5 & 0 & 0 & 0 & 1 & 0 & -1 \\ -2 & 3 & -1 & 0 & 0 & -M & 0 & 1 & 0 \end{bmatrix}$$

```
> B:=pivot(A,3,6);
```

$$B := \begin{bmatrix} 3 & -2 & 1 & 1 & 0 & 0 & 0 & 0 & 5 \\ 1 & 3 & -4 & 0 & 1 & 0 & 0 & 0 & 9 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 6 \\ 0 & -1 & -5 & 0 & 0 & 0 & 1 & 0 & -1 \\ M-2 & M+3 & M-1 & 0 & 0 & 0 & 0 & 1 & 6M \end{bmatrix}$$

```
> C:=pivot(B,2,2);
```

$$C := \begin{bmatrix} \frac{11}{3} & 0 & \frac{-5}{3} & 1 & \frac{2}{3} & 0 & 0 & 0 & 11 \\ 1 & 3 & -4 & 0 & 1 & 0 & 0 & 0 & 9 \\ \frac{2}{3} & 0 & \frac{7}{3} & 0 & \frac{-1}{3} & 1 & 0 & 0 & 3 \\ \frac{1}{3} & 0 & \frac{-19}{3} & 0 & \frac{1}{3} & 0 & 1 & 0 & 2 \\ \frac{2}{3}M-3 & 0 & \frac{7}{3}M+3 & 0 & -\frac{1}{3}M-1 & 0 & 0 & 1 & 3M-9 \end{bmatrix}$$

```
> E:=pivot(C,3,3);
```

$$E := \begin{bmatrix} \frac{29}{7} & 0 & 0 & 1 & \frac{3}{7} & \frac{5}{7} & 0 & 0 & \frac{92}{7} \\ \frac{15}{7} & 3 & 0 & 0 & \frac{3}{7} & \frac{12}{7} & 0 & 0 & \frac{99}{7} \\ \frac{2}{3} & 0 & \frac{7}{3} & 0 & \frac{-1}{3} & 1 & 0 & 0 & 3 \\ \frac{15}{7} & 0 & 0 & 0 & \frac{-4}{7} & \frac{19}{7} & 1 & 0 & \frac{71}{7} \\ \frac{-27}{7} & 0 & 0 & 0 & \frac{-4}{7} & -M-\frac{9}{7} & 0 & 1 & \frac{-90}{7} \end{bmatrix}$$

## 4.4 A Partial Proof

As in the case of the one-phase algorithm, it is necessary to prove certain claims made in the description of the process. We shall not cope with the problem of cycling, but, referring to Fig. 4.1, will prove that

1. the claim of Box A is true; and

2. the claim of Box B is true.

This will show that, apart from cycling, the treatment of each bad row either comes to a stop at Box A or the bad row is turned into a good row. So, after a finite number of steps, either the process stops or it enters the one-phase algorithm when all the bad rows have been eliminated. Apart from the difficulties of cycling, this establishes that the process is a genuine algorithm.

**4.7. Theorem.** *In a two-phase algorithm, if $u_{q+1} < 0$ and $y_{q+1,s} \geq 0$ for all $s$ then there is no feasible solution.*

*Proof.* Our methodology carries with it a proof of this result. We are trying at this stage to increase $u_{q+1}$ to make it positive, and have concluded the the one-phase process has converged, and that the objective function of this problem, namely $u_{q+1}$, is as large as possible. Since in any feasible solution it must be positive, there can be no feasible solutions. □

**4.8. Theorem.** *If $u_{q+1} < 0$ and $y_{q+1,s} < 0$ and there is no $r$ with $1 \leq r \leq q$ such that $y_{rs} > 0$, then pivoting at $y_{q+1,s}$ turns this bad row into a good row and all the good rows above remain as good rows.*

*Proof.* We carry out the calculation as described in the theorem. The manipulations are shown in Table 4.7. We see that after performing the pivot, all the good rows stay good, since we are given that $y_{is} < 0$ for every good row $i$, while the first bad row, about which we pivot, becomes good, since each of the terms in the quotient $u_{q+1}/y_{q+1,s}$ is negative. □

| | $\mathbf{a}_1$ | $\mathbf{a}_2$ | ... | $\mathbf{a}_s$ | ... | $\mathbf{a}_n$ | $\mathbf{e}$ | $\mathbf{b}$ | |
|---|---|---|---|---|---|---|---|---|---|
| $\mathbf{a}_{h_1}$ | | | | $y_{1s}$ | | | 0 | $u_1$ | $(\geq 0)$ |
| $\vdots$ | | | | $\vdots$ | | | $\vdots$ | $\vdots$ | |
| $\mathbf{a}_{h_i}$ | | | | $y_{is}$ | | | 0 | $u_i$ | $(\geq 0)$ |
| $\vdots$ | | | | $\vdots$ | | | $\vdots$ | $\vdots$ | |
| $\mathbf{a}_{h_q}$ | | | | $y_{qs}$ | | | 0 | $u_q$ | $(\geq 0)$ |
| $\mathbf{a}_{h_{q+1}}$ | | | | $y_{q+1,s}$ | | | 0 | $u_{q+1}$ | $(< 0)$ |
| $\vdots$ | | | | $\vdots$ | | | $\vdots$ | $\vdots$ | |
| $\mathbf{a}_{h_m}$ | | | | $y_{ms}$ | | | 0 | $u_m$ | $(< 0)$ |
| $\mathbf{e}$ | $v_1$ | $v_2$ | ... | $v_s$ | ... | $v_n$ | 1 | $z$ | |
| $\mathbf{a}_{h_1}$ | | | | 0 | | | 0 | $u_1 - y_{1,s}(u_{q+1}/y_{q+1,s})$ | $(\geq 0)$ |
| $\vdots$ | | | | $\vdots$ | | | $\vdots$ | $\vdots$ | |
| $\mathbf{a}_{h_i}$ | | | | 0 | | | 0 | $u_i - y_{i,s}(u_{q+1}/y_{q+1,s})$ | $(\geq 0)$ |
| $\vdots$ | | | | $\vdots$ | | | $\vdots$ | $\vdots$ | |
| $\mathbf{a}_{h_q}$ | | | | 0 | | | 0 | $u_q - y_{q,s}(u_{q+1}/y_{q+1,s})$ | $(\geq 0)$ |
| $\mathbf{a}_{h_{q+1}}$ | | | | 1 | | | 0 | $u_{q+1}/y_{q+1,s}$ | $(> 0)$ |
| $\vdots$ | | | | $\vdots$ | | | $\vdots$ | $\vdots$ | |
| $\mathbf{a}_{h_m}$ | | | | 0 | | | 0 | $*$ | (?) |
| $\mathbf{e}$ | $v'_1$ | $v'_2$ | ... | 0 | ... | $v'_n$ | 1 | ? | |

Table 4.7: Pivoting on a bad row; we don't care about $v'_1$, $v'_2$, ..., $v'_n$ at this stage.

## 4.5 Avoiding Non-Negativity

We can show that, given a linear programming problem in which not all the variables are required to be non-negative, it is possible to form a related problem with non-negativity restrictions, whose solution yields the solution of the original problem. If $x_j$ is any such **free** variable (i.e. without non-negativity restriction), the method is to write $x_j = x'_j - x''_j$, with $x'_j$ and $x''_j$ satisfying non-negativity restrictions $x'_j \geq 0$, $x''_j \geq 0$. This gives a related problem with more variables but having non-negativity restrictions.

*4.9. Example.* Let $P$ be the linear programming problem: maximise $3x_1 + x_2$ subject to

$$\begin{aligned} x_1 - 3x_2 &\geq -3, \\ 2x_1 + 3x_2 &\geq -6, \\ 2x_1 + x_2 &\leq 8, \\ 4x_1 - x_2 &\leq 16. \end{aligned}$$

*Solution* There are no non-negativity conditions so we write $x_1 = x'_1 - x''_1$ and $x_2 = x'_2 - x''_2$ and obtain the related problem $P'$:

Maximise $3x_1' - 3x_1'' + x_2' - x_2''$ subject to

$$\begin{aligned} x_1' - x_1'' - 3x_2' + 3x_2'' &\geq -3, \\ 2x_1' - 2x_1'' + 3x_2' - 3x_2'' &\geq -6, \\ 2x_1' - 2x_1'' + x_2' - x_2'' &\leq 8, \\ 4x_1' - 4x_1'' - x_2' + x_2'' &\leq 16, \\ x_1', x_1'', x_2', x_2'' &\geq 0. \end{aligned}$$

It isn't hard to see that when this related problem $P'$ is constructed we have:

1. for each solution of $P$ with objective value $z$ there exists a solution of $P'$ with the same objective value; and

2. for each solution of $P'$ with objective value $z$ there corresponds a unique solution of $P$, which has the same objective value.

There is some abuse of terminology because relating to $P$ we strictly speaking do not have 'feasible' solutions. Nevertheless it follows from (i) and (ii) that if $P$ has an optimal solution then so does $P'$, and conversely, the optimal value being the same, and if $P$ has no optimal solution then neither does $P'$, and conversely.

Here is a MAPLE solution to show how the pivoting goes:

```
> with(linalg): A:=matrix([[-1,1,3,-3],[-2,2,-3,3],
> [2,-2,1,-1],[4,-4,-1,1],[-3,3,-1,1]]):
> b:=vector([3,6,8,16,0]): A1:=concat(A,diag(1,1,1,1,1),b);
```

$$A1 := \begin{bmatrix} -1 & 1 & 3 & -3 & 1 & 0 & 0 & 0 & 0 & 3 \\ -2 & 2 & -3 & 3 & 0 & 1 & 0 & 0 & 0 & 6 \\ 2 & -2 & 1 & -1 & 0 & 0 & 1 & 0 & 0 & 8 \\ 4 & -4 & -1 & 1 & 0 & 0 & 0 & 1 & 0 & 16 \\ -3 & 3 & -1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

```
> A2:=mulrow(A1,3,1/2):A2:=pivot(A2,3,1);
```

$$A2 := \begin{bmatrix} 0 & 0 & \frac{7}{2} & \frac{-7}{2} & 1 & 0 & \frac{1}{2} & 0 & 0 & 7 \\ 0 & 0 & -2 & 2 & 0 & 1 & 1 & 0 & 0 & 14 \\ 1 & -1 & \frac{1}{2} & \frac{-1}{2} & 0 & 0 & \frac{1}{2} & 0 & 0 & 4 \\ 0 & 0 & -3 & 3 & 0 & 0 & -2 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{-1}{2} & 0 & 0 & \frac{3}{2} & 0 & 1 & 12 \end{bmatrix}$$

```
> A3:=mulrow(A2,4,1/3):A3:=pivot(A3,4,4);
```

$$A3 := \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & \frac{-11}{6} & \frac{7}{6} & 0 & 7 \\ 0 & 0 & 0 & 0 & 0 & 1 & \frac{7}{3} & \frac{-2}{3} & 0 & 14 \\ 1 & -1 & 0 & 0 & 0 & 0 & \frac{1}{6} & \frac{1}{6} & 0 & 4 \\ 0 & 0 & -1 & 1 & 0 & 0 & \frac{-2}{3} & \frac{1}{3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{7}{6} & \frac{1}{6} & 1 & 12 \end{bmatrix}$$

Thus we have $x_1' = 4$, $x_2'' = 0$, $x_3 = 7$ and $x_4 = 14$. The remaining variables, $x_1''$, $x_2'$, $x_5$ and $x_6$ are not basic variables and so are zero. The maximum value of the objective function is 12. In terms of the original variables, this is attained when $x_1 = 4$ and $x_2 = 0$.

**Other Substitutions** A similar use of substitution can be made if, for example, a problem has, instead of non-negativity restrictions, other restrictions like $x_1 \geq 2$, $x_2 \leq 6$, $x_3 \geq -4$. One then substitutes $x_1' = x_1 - 2$, $x_2' = 6 - x_2$, $x_3' = x_3 + 4$ for then $x_1', x_2', x_3' \geq 0$.

## 4.6 An Alternative View

We finish this chapter by showing an alternative way of reducing problems to the one-phase algorithm which makes greater use of artificial variables. This is the approach taken by Press et al. (1992) to the two phase algorithm. Again we illustrate with an example.

*4.10. Example.* Maximise $z = x_1 + x_2 + 3x_3 - \frac{1}{2}x_4$ subject to $x_i \geq 0$ for $1 \leq i \leq 4$ and

$$\begin{aligned} x_1 + 2x_3 &\leq 740, \\ 2x_2 - 7x_4 &\leq 0, \\ x_2 - x_3 + 2x_4 &\geq \tfrac{1}{2}, \\ x_1 + x_2 + x_3 + x_4 &= 9. \end{aligned}$$

To help keep count, I'm going to use $y_i$ for a slack variable and $z_i$ for an artificial variable. We first introduce slack variables to give

$$\begin{aligned} x_1 + 2x_3 + y_1 &= 740, \\ 2x_2 - 7x_4 + y_2 &= 0, \\ x_2 - x_3 + 2x_4 - y_3 &= \tfrac{1}{2}, \\ x_1 + x_2 + x_3 + x_4 &= 9. \end{aligned}$$

Of course we only need three slack variables, and the third constraint shows we are going to need the two-phase version of the algorithm. Now comes the alternative approach to artificial variables — lets use *lots!*. Specifically we define an artificial variable for *each*

constraint by

$$\begin{aligned} z_1 &= 740 - x_1 - 2x_2 - y_1, \\ z_2 &= -2x_2 + 7x_4 - y_2, \\ z_3 &= \tfrac{1}{2} - x_2 + x_3 - 2x_4 + y_3, \\ z_4 &= 9 - x_1 - x_2 - x_3 - x_4. \end{aligned}$$

Thus we have introduced a "full set" of artificial variables $z_1 \geq 0$, $z_2 \geq 0$, $z_3 \geq 0$ and $z_4 \geq 0$. We chose the sign of the right hand side to ensure that each constant is positive. Thus we could write the above system in tableau form; we have then chosen things so this is a one-phase problem; specifically there is a feasible solution with only the artificial variables non-zero.

However this isn't the same system as the original unless each $z_i$ is zero. Note that $z' = z_1 + z_2 + z_3 + z_4 \geq 0 + 0 + 0 + 0 \geq 0$. Thus the minimum value of $z'$ is non-negative. Note also that if $z' = 0$ then each $z_i = 0$.

Thus *if* we can find a basic feasible solution to this new problem, in which $z' = 0$, then all our artificial variables vanish, and we can go to work on the "proper" variables, and the "real" objective function exactly as we did in the third tableau of Example 4.6. So we start by working on an *auxiliary* objective function $z' = z_1 + z_2 + z_3 + z_4$ which we seek to minimise. On the way, we transform the real objective function during each pivot, thus ensuring our new system remains row-equivalent to the old. Considering the auxiliary objective function then gives an alternative to the use of "bad rows" as a way of finding a basic feasible solution to our original problem.

Notice also that if our original problem *has* a basic feasible solution, then that solution gives the correct minimum value for the auxiliary objective function. So the above method will succeed providing the original problem has a solution.

To summarise this method:

- introduce slack variables so each constraint is an equality;
- introduce artificial variables with signs chosen so the new problem has a basic feasible solution;
- minimise the auxiliary objective function; and
- use the resulting basic feasible solution of the original problem to start the one-phase algorithm.
- Finally note that if the auxiliary objective function cannot be reduced to zero, the original problem has no solution.

*4.11. Example.* Use MAPLE to solve the problem in Example 4.10. Try it using the minimum number of artificial variables as described in Section 4.3 and also using a "full set" of them as just described. Summarise the advantages and disadvantages of each method.

## 4.7 Questions 3 (Hints and solutions start on page 112.)

*4.1. Q.* a) In carrying out the two-phase simplex algorithm, assume you arrive at the following tableau, with one bad row and $q$ good rows.

| Basis | $\mathbf{a}_1$ | $\dots$ | $\mathbf{a}_s$ | $\dots$ | $\mathbf{a}_n$ | $\mathbf{e}$ | $\mathbf{b}$ |
|---|---|---|---|---|---|---|---|
| $\mathbf{a}_{h_1}$ | $y_{1,1}$ | $\dots$ | $y_{1,s}$ | $\dots$ | $y_{1,n}$ | 0 | $u_1 \quad (> 0)$ |
| $\vdots$ | | | $\vdots$ | | | $\vdots$ | $\vdots$ |
| $\mathbf{a}_{h_q}$ | | | $y_{q,s}$ | | | 0 | $u_q \quad (> 0)$ |
| $\mathbf{a}_{h_{q+1}}$ | $y_{q+1,1}$ | $\dots$ | $y_{q+1,s}$ | $\dots$ | $y_{q+1,n}$ | 0 | $u_{q+1} \quad (< 0)$ |
| $\mathbf{e}$ | $u_1$ | $\dots$ | $u_s$ | $\dots$ | $u_n$ | 1 | $z$ |

The next step in the algorithm forces $\mathbf{a_s}$ to be introduced into the basis. Given that $y_{i,s} \leq 0$ for $1 \leq i \leq q$, and that $y_{q+1,s} < 0$, about which entry in the column should you pivot? Prove that your choice leads to a new tableau corresponding to a basic feasible solution.

b) Consider the linear programming problem:

minimise $6x_1 + 7x_2 + x_3 + 15x_4$ subject to $x_1 \geq 0$, $x_2 \geq 0$, $x_3 \geq 0$, $x_4 \geq 0$ and

$$x_1 + x_2 + 3x_4 = 6,$$
$$x_1 + x_2 - x_3 + 2x_4 \geq 5,$$

Apply the simplex method to show that the optimal value is 33, and give values of the variables at which this is attained. Describe briefly the reasoning behind each step. [When you have a free choice of pivots in a row, you are recommended to choose the one furthest to the left.]

*4.2. Q.* Explain how the simplex method, when applied to an unbounded linear programming problem, will eventually reveal the fact that it is unbounded.

Use the Simplex Algorithm to minimise $-6x_1 + 2x_2 - 3x_3$, subject to $x_1, x_2, x_3 \geq 0$ and

$$x_1 + x_2 - 2x_3 \leq 2,$$
$$-4x_1 + x_2 - x_3 \geq 6,$$
$$x_1 + 3x_2 - 8x_3 \leq 8.$$

Describe briefly the reasoning behind each step.

*4.3. Q.* a) A small portion of a tableau giving a basic feasible solution of a linear programming problem is shown in the following table and the simplex algorithm indicates the need to pivot about the entry $y_{rs}$.

| Basis | $\dots$ | $\mathbf{a}_s$ | $\dots$ | $\mathbf{b}$ |
|---|---|---|---|---|
| $\vdots$ | | $\vdots$ | | $\vdots$ |
| $\mathbf{a}_{h_i}$ | $\dots$ | $y_{is}$ | $\dots$ | $u_i$ |
| $\vdots$ | | $\vdots$ | | $\vdots$ |
| $\mathbf{a}_{h_r}$ | $\dots$ | $y_{rs}$ | $\dots$ | $u_r$ |
| $\vdots$ | | $\vdots$ | | $\vdots$ |

By performing this pivot on the entries shown, giving the resulting partial tableau, explain why it was necessary to consider certain ratios before choosing to pivot in row $r$.

b) Use the Simplex Algorithm to minimise $-2x_1 + x_2 - 4x_3$, subject to $x_1, x_2, x_3 \geq 0$ and

$$\begin{array}{rcrcrl} x_1 & + & & & x_3 & \leq 4, \\ x_1 & - & 2x_2 & - & 3x_3 & = 2, \\ x_1 & - & 3x_2 & - & x_3 & \leq 1. \end{array}$$

Describe briefly the reasoning behind each step.

*4.4. Q.* Use the Simplex Algorithm to minimise $6x_1 - x_2 + 12x_3$, subject to $x_1, x_2, x_3 \geq 0$ and

$$\begin{array}{rcrcrl} x_1 & + & & & 3x_3 & \leq 2, \\ 2x_1 & - & x_2 & - & x_3 & = 3, \\ 4x_1 & - & x_2 & - & 2x_3 & \geq 7. \end{array}$$

Describe briefly the reasoning behind each step.

# Chapter 5

# Duality

## 5.1 Formulation of the Dual

**5.1. Definition.** Given a Linear Programming Problem (called the **primal** problem) in the form: maximise $c_1x_1 + c_2x_2 + \ldots + c_nx_n$ subject to

$$\begin{aligned} \sum_{j=1}^{n} a_{ij}x_j &\leq b_i \text{ for } i = 1, 2, \ldots, m, \\ x_j &\geq 0 \text{ for } j = 1, 2, \ldots, n. \end{aligned}$$

then its **dual** is the problem: minimise $b_1w_1 + b_2w_2 + \ldots + b_mw_m$ subject to

$$\begin{aligned} \sum_{j=1}^{m} a_{ji}w_j &\geq c_i \text{ for } i = 1, 2, \ldots, n, \\ w_j &\geq 0 \text{ for } j = 1, 2, \ldots, m. \end{aligned}$$

In matrix notation, the primal can be written as: maximise $\mathbf{c}^T\mathbf{x}$ subject to $A\mathbf{x} \leq \mathbf{b}$, $\mathbf{x} \geq \mathbf{0}$. and the dual as: minimise $\mathbf{b}^T\mathbf{w}$ subject to $A^T\mathbf{w} \geq \mathbf{c}, \mathbf{w} \geq \mathbf{0}$.

It is clear that any Linear Programming Problem can be put in the form of a primal problem. Thus a minimising problem can be reformulated as a maximising problem by changing the sign of the objective function and any $\geq$ inequalities can be reformulated as $\leq$ inequalities.

There is a particular way of dealing with constraints that are equations. Certainly any equation is equivalent to two inequalities which can both be written as $\leq$ inequalities. In the resulting dual one finds that the two corresponding variables can be replaced by one variable, which however is unrestricted. Thus, if the $i^{\text{th}}$ constraint in the primal (before reformulation) is an equation then the $i^{\text{th}}$ variable in the dual (after reformulation in the way illustrated below) is unrestricted.

*5.2. Example.* Find the dual of the problem: maximise $10x_1 + x_2 + 4x_3 + 7x_4$ subject to

$$\begin{aligned} 3x_1 - 2x_2 + 7x_3 - 6x_4 &\leq 5, \\ 8x_1 + 4x_2 - 11x_3 + x_4 &\geq 12, \\ 9x_1 + 15x_2 + 14x_3 + 10x_4 &= 13, \\ x_1, x_2, x_3, x_4 &\geq 0. \end{aligned}$$

*Solution* Putting this in the required form gives the primal:
maximise $10x_1 + x_2 + 4x_3 + 7x_4$ subject to

$$\begin{aligned} 3x_1 - 2x_2 + 7x_3 - 6x_4 &\leq 5, \\ -8x_1 - 4x_2 + 11x_3 - x_4 &\leq -12, \\ 9x_1 + 15x_2 + 14x_3 + 10x_4 &\leq 13, \\ -9x_1 - 15x_2 - 14x_3 - 10x_4 &\leq -13, \\ x_1, x_2, x_3, x_4 &\geq 0. \end{aligned}$$

The dual of this problem is: minimise $5w_1 - 12w_2 + 13w_3' - 13w_3''$ subject to

$$\begin{aligned} 3w_1 - 8w_2 + 9w_3' - 9w_3'' &\geq 10, \\ -2w_1 - 4w_2 + 15w_3' - 15w_3'' &\geq 1, \\ 7w_1 + 11w_2 + 14w_3' - 14w_3'' &\geq 4, \\ -6w_1 - w_2 + 10w_3' - 10w_3'' &\geq 7, \\ w_1, w_2, w_3', w_3'' &\geq 0. \end{aligned}$$

Now let $w_3 = w_3' - w_3''$ and the problem becomes: minimise $5w_1 - 12w_2 + 13w_3$ subject to

$$\begin{aligned} 3w_1 - 8w_2 + 9w_3 &\geq 10, \\ -2w_1 - 4w_2 + 15w_3 &\geq 1, \\ 7w_1 + 11w_2 + 14w_3 &\geq 4, \\ -6w_1 - w_2 + 10w_3 &\geq 7, \\ w_1, w_2 &\geq 0, \end{aligned}$$

noting that $w_3$, corresponding to the third of the original constraints which was an equation, is unrestricted.

**5.3. Theorem.** *The dual of the dual is the primal.*

*Proof.* This is simply a matter of chasing the definitions, but is worthwhile because it gives familiarity with the concepts. We write the primal as:

$$\text{maximise } \mathbf{c}^T\mathbf{x} \text{ subject to } A\mathbf{x} \leq \mathbf{b} \text{ and } \mathbf{x} \geq \mathbf{0},$$

and so the dual is:

$$\text{minimise } \mathbf{b}^T\mathbf{w} \text{ subject to } A^T\mathbf{w} \geq \mathbf{c} \text{ and } \mathbf{w} \geq \mathbf{0}.$$

We now have to reformulate this as a primal problem, and so write it as a maximising problem:

$$\text{maximise } (-\mathbf{b})^T\mathbf{w} \text{ subject to } (-A)^T\mathbf{w} \leq -\mathbf{c} \text{ and } \mathbf{w} \geq \mathbf{0}.$$

In this form we can again write down a dual; we call it the bidual, because it is the dual of the (reformulated) dual. The bidual is:

$$\text{minimise } (-\mathbf{c})^T\mathbf{x} \text{ subject to} \big((-A)^T\big)^T\mathbf{x} \geq \big((-\mathbf{b})^T\big)^T \text{ and } \mathbf{x} \geq \mathbf{0}.$$

Rewriting this to remove double transposes, and negative signs gives

$$\text{maximise } \mathbf{c}^T\mathbf{x} \text{ subject to } A\mathbf{x} \leq \mathbf{b} \text{ and } \mathbf{x} \geq \mathbf{0},$$

which is the original problem. □

A consequence of the theorem is that, if desired, the primal can be formulated in the form: minimise $\mathbf{c}^T\mathbf{x}$ subject to $A\mathbf{x} \geq \mathbf{b}, \mathbf{x} \geq \mathbf{0}$, in which case its dual is the problem: maximise $\mathbf{b}^T\mathbf{w}$ subject to $\mathbf{A}^T\mathbf{w} \leq \mathbf{c}$, $\mathbf{w} \geq \mathbf{0}$. In other words, we can easily confuse the rôles of maximising and minimising, without running into confusion.[1]

It may also be useful to note that given a problem in which all the variables are unrestricted, one can use the technique above in reverse to obtain as its dual a problem in which all the constraints are equations. We have already seen a technique for solving such problems, and we are next going to show that solutions of the primal and dual problems are related.

## 5.2 The Fundamental Theorem

**5.4. Lemma.** *Let* $\mathbf{x}$ *and* $\mathbf{w}$ *be feasible solutions of the above primal problem and its dual. Then* $\mathbf{c}^T\mathbf{x} \leq \mathbf{b}^T\mathbf{w}$.

*Proof.* This result is a simple symbol chase from the hypothesis: since $\mathbf{x}$ and $\mathbf{w}$ are feasible solutions of the above primal problem and its dual, with the notation we have been using as standard, we know that

$$\sum_{j=1}^{n} a_{ij}x_j \leq b_i \text{ for } i = 1, 2, \ldots, m, \quad \text{and} \quad \sum_{i=1}^{m} a_{ij}w_i \geq c_i \text{ for } i = 1, 2, \ldots, n.$$

Computing, we thus have

$$\begin{aligned}
\sum_{i=1}^{m} b_i w_i &\geq \sum_{i=1}^{m} \left( \sum_{j=1}^{n} a_{ij}x_j \right) w_i \\
&= \sum_{j=1}^{n} \left( \sum_{i=1}^{m} a_{ij}w_i \right) x_j \quad \text{interchanging the order of summation,} \\
&\geq \sum_{j=1}^{n} c_j x_j.
\end{aligned}$$

which is the result claimed. □

You may prefer the following vector version of this — it is the same argument expressed in a different way. Note first that if $\mathbf{u} \leq \mathbf{v}$ and $\mathbf{x} \geq \mathbf{0}$ then $\mathbf{x}^T\mathbf{u} \leq \mathbf{x}^T\mathbf{v}$ since we first multiply an inequality by $x_j \geq 0$ and then add. Then, since $\mathbf{c} \leq A^T\mathbf{w}$, and $\mathbf{x} \geq \mathbf{0}$, we have

$$\mathbf{x}^T\mathbf{c} \leq \mathbf{x}^T A^T\mathbf{w} = (A\mathbf{x})^T\mathbf{w} \leq \mathbf{b}^T\mathbf{w}$$

where in the last inequality we have used the note above and the fact that $A\mathbf{x} \leq \mathbf{b}$. Since $\mathbf{x}^T\mathbf{c} = \mathbf{c}^T\mathbf{x}$, the result follows.

[1] I hope you aren't confused!

**5.5. Lemma.** *Let* $\mathbf{x}$ *and* $\mathbf{w}$ *be feasible solutions of the above primal problem and its dual. If* $\mathbf{c}^T\mathbf{x} = \mathbf{b}^T\mathbf{w}$ *then* $\mathbf{x}$ *and* $\mathbf{w}$ *are* optimal *solutions of the primal and dual.*

*Proof.* This is a simple consequence of Lemma 5.4. Recall that for the primal problem, we are trying to maximise

$$\sum_{j=1}^{n} c_j x_j \leq \sum_{i=1}^{m} b_i w_i,$$

where we have used Lemma 5.4 to get the inequality. If the above inequality is actually an equality, then however we change any $x_j$ cannot increase the value of the objective function $\mathbf{c}^T\mathbf{x}$. And in the same way, however we change any $w_i$ cannot decrease the value of the objective function $\mathbf{b}^T\mathbf{w}$ of the dual problem. So both must already correspond to optimal solutions; indeed we see that both the primal and the dual problem have optimal solutions with the *same* optimal value. □

**5.6. Theorem.** *If a Linear Programming Problem has an optimal solution then its dual has an optimal solution, with the same optimal value.*

*Proof.* We start with the primal:-

$$\text{maximise } \mathbf{c}^T\mathbf{x} \text{ subject to } A\mathbf{x} \leq \mathbf{b} \text{ and } \mathbf{x} \geq \mathbf{0},$$

and we assume the primal problem has an optimal solution $\mathbf{x}$, with optimal value $z = \mathbf{c}^T\mathbf{x}$. Then the initial tableau is as shown in Table 5.1. Note that the last row uses the fact that we know the coefficients of the objective function; we are also relying on the use of the techniques we have already discussed (including the use of artificial variables) to introduce the correct number (a total of $m$) of slack variables.

| Basis | $\mathbf{a}_1$ | $\mathbf{a}_2$ | ... | $\mathbf{a}_n$ | Slack Variables | | | $\mathbf{e}$ | $\mathbf{b}$ |
|---|---|---|---|---|---|---|---|---|---|
| Last | | | | | | | | 0 | $b_1$ |
| $m$ | | | | | | | | 0 | $b_2$ |
| $\vdots$ | | | | | | $\mathbf{I}_m$ | | $\vdots$ | $\vdots$ |
| columns | | | | | | | | 0 | $b_m$ |
| $\mathbf{e}$ | $-c_1$ | $-c_2$ | ... | $-c_n$ | 0 | ... | 0 | 1 | 0 |

Table 5.1: Initial tableau for the primal $\mathbf{Ax} = \mathbf{b}$.

Now consider the situation when the simplex algorithm has terminated to give the final tableau shown in Table 5.2. Since is *is* a final tableau, none of the $v_1,\ldots,v_n,w_1,\ldots,w_m$ has the proper sign for improvement, and so $v_j \geq 0$ and $w_i \geq 0$ for all $i, j$.

Consider next the way in which the final tableau is obtained from the initial tableau; it is simply by a finite number of consecutive pivoting operations, on rows *excluding* the last row. This means that the last row in the final tableau can be written as a linear combination of the rows in the initial tableau of the form

$$\begin{aligned} \text{Last row of final tableau} &= \text{Last row of initial tableau} \\ &+ \lambda_1 \times \text{first row of initial tableau} + \ldots + \\ &+ \lambda_m \times m^{\text{th}} \text{ row of initial tableau.} \end{aligned}$$

| Basis | $\mathbf{a}_1$ | $\mathbf{a}_2$ | ... | $\mathbf{a}_n$ | Slack Variables | | | $\mathbf{e}$ | $\mathbf{b}$ |
|---|---|---|---|---|---|---|---|---|---|
| Basic | | | | | | | | 0 | $u_1$ |
| $m$ | | | | | | | | 0 | $u_2$ |
| $\vdots$ | | | | | | $\vdots$ | | $\vdots$ | $\vdots$ |
| columns | | | | | | | | 0 | $u_m$ |
| $\mathbf{e}$ | $v_1$ | $v_2$ | ... | $v_n$ | $w_1$ | ... | $w_m$ | 1 | $z$ |

Table 5.2: Final tableau for the primal $\mathbf{Ax} = \mathbf{b}$.

Now note that the initial tableau has $\mathbf{I}_m$ in the last $m$ columns of the body of the tableau, and so we can read off the relevant values of $\lambda$; we have $\lambda_1 = w_1$, $\lambda_2 = w_2$, ..., $\lambda_m = w_m$. Thus we can compute each $v_j$; we have

$$v_j = -c_j + w_1 a_{1j} + \ldots + w_m a_{mj} \qquad (j = 1, \ldots, n) \quad \text{and } z = \sum_{i=1}^{m} w_i b_i.$$

In vector form, we thus have that $z = \mathbf{b}^T\mathbf{w}$, and $\mathbf{A}^T\mathbf{w} = \mathbf{v} + \mathbf{c} \geq \mathbf{c}$, since $\mathbf{v} \geq 0$; and also $\mathbf{w} \geq 0$. This shows that $\mathbf{w}$ is a feasible solution of the dual problem, and that $\mathbf{c}^T\mathbf{x} = z = \mathbf{b}^T\mathbf{w}$. Thus by Lemma 5.5, both $\mathbf{x}$ and $\mathbf{w}$ are optimal. □

**5.7. Corollary.** *If a Linear Programming Problem and its dual both have feasible solutions then they both have optimal solutions.*

*Proof.* By hypothesis, we have feasible solutions of the primal and dual problems respectively, say $\mathbf{x}$ and $\mathbf{w}$. Then by Lemma 5.4, $\mathbf{c}^T\mathbf{x} \leq \mathbf{b}^T\mathbf{w}$. Thus $\mathbf{c}^T\mathbf{x}$ is bounded above, and so when we apply the simplex algorithm, it does not terminate in box B of the one-phase algorithm, and as such is guaranteed to terminate with an optimal solution. Then by Theorem 5.6, both the primal and dual problems have an optimal solution. □

To complete the possibilities, we note that the following cases can arise.

1. The primal and dual may both have no feasible solutions. Consider the primal Linear Programming Problem: maximise $x_1 + x_2 + x_3$ subject to

$$\begin{aligned} x_1 + x_2 - x_3 &\leq 1, \\ x_1 - x_2 + x_3 &\leq -2, \\ x_1, x_2, x_3 &\geq 0, \end{aligned}$$

   and its dual.

2. It can be that the primal has no feasible solutions but the dual does have feasible solutions. Consider the primal Linear Programming Problem: maximise $x_1$ subject to $x_1 \leq 1, -x_1 \leq -2, x_1 \geq 0$ and its dual.

The dual problem may be worth considering in cases where the primal problem is difficult to solve. For example, the primal may require the two-phase algorithm and the

dual may not. But the notion of dual is important because where a Linear Programming Problem arises from a practical problem the dual may have some interpretation that it is useful to appreciate.

*5.8. Example.* We work the first of these results as an example.

*Solution* In the primal problem an attempt to turn the bad row into a good row fails; the best we can do is raise the value of the constant to $-1$, so there is no feasible solution of the primal problem. The tableau is given in Table 5.3

| Basis | $\mathbf{a}_1$ | $\mathbf{a}_2$ | $\mathbf{a}_3$ | $\mathbf{a}_4$ | $\mathbf{a}_5$ | $\mathbf{e}$ | $\mathbf{b}$ |
|---|---|---|---|---|---|---|---|
| $\mathbf{a}_4$ | 1 | 1 | -1 | 1 | 0 | 0 | 1 |
| $\mathbf{a}_5$ | 1 | -1 | 1 | 0 | 1 | 0 | -2 |
| $\mathbf{e}$ | -1 | -1 | -1 | 0 | 0 | 1 | 0 |
| $\mathbf{a}_4$ | 1 | 1 | -1 | 1 | 0 | 0 | 1 |
| $\mathbf{a}_5$ | 2 | 0 | 0 | 0 | 1 | 0 | -1 |
| $\mathbf{e}$ | 0 | 0 | -2 | 1 | 0 | 1 | 0 |

Table 5.3: Duality example: the primal problem has no optimal solution.

The primal problem has

$$\mathbf{c} = (1, 1, 1), \qquad \mathbf{A} = \begin{pmatrix} 1 & 1 & -1 \\ 1 & -1 & 1 \end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix} 1 \\ -2 \end{pmatrix},$$

and so the dual becomes: minimise $w_1 - 2w_2$ subject to

$$\begin{aligned} w_1 + w_2 &\geq 1, \\ w_1 - w_2 &\geq 1, \\ -w_1 + w_2 &\geq 1, \\ w_1, w_2, w_3 &\geq 0. \end{aligned}$$

In this case it is clear that there is no feasible solution, since the last constraint can be written as $w_1 - w_2 \leq -1$, which is clearly incompatible with the second constraint.

## 5.3 Shadow Prices

Often the dual of a Linear Programming Problem has a natural interpretation. We illustrate this with an example.

A poor student has decided that she needs to obtain each day an additional $b_i$ grams of vitamin $V_i$ for $1 \leq i \leq m$ by consuming additional quantities of a number of foods $F_j$ for $1 \leq j \leq n$. Food $F_j$ contains a total of $a_{ij}$ grams of vitamin $V_i$ and costs £$c_j$ per gram. These values are given for each $i$ and $j$. Our first problem is to determine the amount, in grams, of each food to be consumed in order to provide the required additional vitamins at the least cost.

To this end, let $x_j$ be the amount, in grams, of food $F_j$ to be consumed; clearly we have $\mathbf{x} \geq 0$. The total cost, which we want to minimise, is then $\mathbf{c}^T\mathbf{x}$. With this choice,

we consume $\sum_{j=1}^{n} a_{ij}x_j$ grams of vitamin $V_i$, so we meet the required intake provided that $\mathbf{Ax} \geq \mathbf{b}$.

At this point a wicked salesman shows up with a stock of vitamin pills, and rather flexible prices. He wishes to persuade our poor student *not* to get her extra vitamins from real food, but instead to buy them directly in the form of vitamin pills. For each $1 \leq i \leq m$, pill $P_i$ contains one gram of $V_i$. What price should the salesman charge for $P_i$ in order that our student abandon her healthy lifestyle in favour of a cheaper solution, and yet the salesman still maximises his profit?

If the salesman charges £$w_i$ for each $P_i$, his total income will be $\mathbf{w}^T\mathbf{b}$, which we seek to maximise.[2] Clearly $\mathbf{w} \geq 0$. What other constraints are there on $\mathbf{w}$? One way to express it is that the cost of pills to supply the vitamin content of 1 gram of $V_i$ should be less than the cost of the food. Since $F_j$ supplies $a_{ij}$ grams of vitamin $V_i$, we must have

$$\sum_{i=1}^{m} a_{ij}w_i \leq c_j \quad \text{for} \quad 1 \leq j \leq n$$

or $\mathbf{A}^T\mathbf{w} \leq \mathbf{c}$.

In other words, the salesman's problem is dual to that of the student's.

Let $z^*$ be the optimal value; the minimum of $\mathbf{c}^T\mathbf{x}$. We can consider $z^*$ as a function of the given $a_{ij}$, $b_i$ and $c_j$. Then if

$$\frac{\partial z^*}{\partial b_i} = \lambda_i$$

we observe that $\lambda_i$ is the sensitivity of $z$ to changes in $b_i$, or the increase in the objective function if $b_i$ is increased by 1. We call $\lambda_i$ the **shadow price** of the vitamin. The above argument suggest that these give the solution to the dual problem.

## 5.4 The Dual Simplex Method

Although we don't go into details here, there is another reason why duality proves valuable, associated with the need, in real life, to modify the constraints of a problem. Supposed we have already solved (say) a maximising problem. We can describe the solution as feasible (every entry in the last column is non-negative) and optimal (every entry in the bottom row is non-negative). Adding an additional constraint is likely to render the "current" solution infeasible, although it will remain optimal in the above sense. Applying the conventional algorithm will first work on the (potentially) bad row, and then re-do the optimisation. In contrast the dual problem is likely to become non optimal (a bad row "duals" to a proper sign for improvement) but feasible (an optimal objective row duals to a feasible but non optimal solution). As such, restoring optimality is likely in practice to be quicker working on the dual. For more details, see for example Kolman & Beck (1995, Section 3.4)

[2] We ignore the unlikely situation in which he sells a pill for below his cost; the pills are very cheap to make!

## 5.5 Questions 4 (Hints and solutions start on page 118.)

*5.1. Q.* Write down the dual of the linear programming problem given in Question 4.1, and show that the dual can be transformed into a linear programming problem in which one of the variables is unrestricted. Using the information contained in Question 4.1, give the optimal value for the dual, and explain your reasoning. [You are *not* expected to solve the problem.]

*5.2. Q.* a) A linear programming problem, the "primal" problem, is given in the form:

maximise $\mathbf{c}^T\mathbf{x}$ subject to the constraints $\mathbf{A}\mathbf{x} \leq b$ and $\mathbf{x} \geq 0$.

Write down the dual problem. Given that $\mathbf{x}$ and $\mathbf{w}$ are feasible solutions of the above primal and its dual, show that $\mathbf{c}^T\mathbf{x} \leq \mathbf{b}^T\mathbf{w}$. What can you deduce if $\mathbf{c}^T\mathbf{x} = \mathbf{b}^T\mathbf{w}$? Prove your assertion.

b) The following tableau is derived from a linear programming problem of the above form by inserting $x_4$, $x_5$ and $x_6$ as slack variables.

| Basis | $\mathbf{a}_1$ | $\mathbf{a}_2$ | $\mathbf{a}_3$ | $\mathbf{a}_4$ | $\mathbf{a}_5$ | $\mathbf{a}_6$ | $\mathbf{e}$ | $\mathbf{b}$ |
|---|---|---|---|---|---|---|---|---|
| $\mathbf{a}_4$ | 2 | 3 | -1 | 1 | 0 | 0 | 0 | 4 |
| $\mathbf{a}_5$ | -1 | -2 | 0 | 0 | 1 | 0 | 0 | 3 |
| $\mathbf{a}_6$ | -1 | $1^*$ | -2 | 0 | 0 | 1 | 0 | 1 |
| $\mathbf{e}$ | -10 | -20 | 10 | 0 | 0 | 0 | 1 | 0 |

Write down the original problem in "primal" form and hence write down the dual problem. After one pivot in the above tableau, the following tableau is obtained.

| Basis | $\mathbf{a}_1$ | $\mathbf{a}_2$ | $\mathbf{a}_3$ | $\mathbf{a}_4$ | $\mathbf{a}_5$ | $\mathbf{a}_6$ | $\mathbf{e}$ | $\mathbf{b}$ |
|---|---|---|---|---|---|---|---|---|
| $\mathbf{a}_4$ | 5 | 0 | 5 | 1 | 0 | -3 | 0 | 1 |
| $\mathbf{a}_5$ | -3 | 0 | -4 | 0 | 1 | 2 | 0 | 5 |
| $\mathbf{a}_2$ | -1 | 1 | -2 | 0 | 0 | 1 | 0 | 1 |
| $\mathbf{e}$ | -30 | 0 | -30 | 0 | 0 | 20 | 1 | 20 |

Solve the original linear programming problem and also write down a solution of the dual problem.

*5.3. Q.* Let $P$ be the linear programming problem: minimise $8x_1 + 5x_2 + 4x_3$ subject to the constraints $x_1 \geq 0$, $x_2 \geq 0$, $x_3 \geq 0$, and

$$x_1 + 5x_2 - x_3 \geq 1,$$
$$-x_1 + 2x_2 - x_3 \leq 4.$$

Write down the dual problem $P^*$. What can you conclude if both $P$ and $P^*$ have feasible solutions?

# Chapter 6

# The Theory of Games

## 6.1 Matrix Games

A game is, in general terms, a conflict situation between opponents that can be rationally analysed. The class of games we shall study are those in which the possible outcomes can be specified by means of a matrix:

**6.1. Definition.** Let there be two players, $R$ (or Rowman) and $C$ (or Columnman) and let $\mathbf{A} = [a_{ij}]$ be an $m \times n$ matrix. In the **matrix game** associated with $\mathbf{A}$, $R$ chooses a row $\mathbf{a}_i$ and simultaneously $C$ chooses a column $\mathbf{a}^j$. As a result $C$ pays to $R$ an amount of $a_{ij}$ units.

It is a convention that the matrix element $a_{ij}$ gives the amount that $C$ pays to $R$, called the **payoff**. Such a game is a "zero-sum" game in that the total amount received by the two players is zero: $R$ receives $a_{ij}$ and $C$ receives $-a_{ij}$.

*6.2. Example.* Assume that each of Rowman and Columnman put down a £1 coin. The game is such that if both coins come down heads, or both come down tails, then Rowman wins both coins; if they disagree, then Columnman wins both coins. Describe this as a matrix game.

*Solution* Let the table represent the possible outcomes as follows:

| | $H$ | $T$ | ⟵ C's coin |
|---|---|---|---|
| $H$ | 1 | $-1$ | |
| $T$ | $-1$ | 1 | |

↑
R's coin

We have written the value to Rowman in each entry; this is also the loss to Columnman of the given outcome, and the interpretation of the table is exactly as required.

Sometimes a game is easy to analyse. Consider the matrix game

$$\begin{pmatrix} -1 & 1 & 4 \\ -3 & 0 & 1 \end{pmatrix}.$$

Then Rowman is always better off choosing row 1, while Columnman always does better by choosing column 1, and so we expect the game to result in a payoff to Columnman of 1.

But the choices are not always this easy; our aim is to come up with convincing strategies in more general situations.

## 6.2 Pure strategies

When such a game is played just once, the possible strategies are easy to analyse.

**6.3. Definition.** Let $\mathbf{A} = [a_{ij}]$ and let

$$\alpha = \max_i \min_j a_{ij}, \qquad \beta = \min_j \max_i a_{ij}.$$

Then $\alpha$ and $\beta$ are the **lower** and **upper** values of the matrix game $\mathbf{A}$.

It is worth examining briefly what $\alpha$ represents; this is the game as seen by Rowman, who first looks at each row in turn, and finds the smallest entry (which is the worst outcome for him). This then is what he can expect if he takes a pessimistic view of the outcome (or in some circumstances, believes that Columnman is playing skillfully). He then decides to minimise his loss by finding which row has the largest such value; the least bad outcome as far as Rowman is concerned. We see then that Rowman can gain at least $\alpha$, and possibly more. In the same way Columnman must pay $\beta$, but possibly less.

**6.4. Theorem.** *The lower and upper values $\alpha$ and $\beta$ of a matrix game satisfy $\alpha \leq \beta$.*

*Proof.* This is simply a case of using the symbols, and keeping an eye on what is required. The first statement below is trivial, as is the second:

$$\begin{aligned} a_{ij} &\leq a_{ij} \qquad \text{for all } i \text{ and } j, \\ \min_j a_{ij} &\leq a_{ij} \qquad \text{for all } i \text{ and } j, \\ \min_j a_{ij} &\leq \max_i a_{ij} \qquad \text{for all } i \text{ and } j. \end{aligned}$$

At this stage, note that the left hand side does not depend on $j$, while the right hand side does not depend on $i$. Thus

$$\begin{aligned} \max_i \min_j a_{ij} &\leq \min_j \max_i a_{ij} \qquad \text{for all } i \text{ and } j, \\ \alpha &\leq \beta. \end{aligned}$$

This is of course the required result. □

**6.5. Definition.** If $R$ chooses a row $\mathbf{a}_i$ such that $\min_j a_{ij} = \alpha$ then $R$ uses a pure **maxmin** strategy. If $C$ chooses a column $\mathbf{a}^j$ such that $\max_i a_{ij} = \beta$ then $C$ uses a pure **minmax** strategy.

These strategies can be called **conservative** strategies. Player $R$ is making certain that, of all the possible smallest winnings he can receive, he receives the largest. Whereas $C$ is making certain that, of all the possible largest payments he may have to make, he pays out the least.

*6.6. Example.* Let $\mathbf{A} = \begin{pmatrix} 0 & 1 & -3 & 4 \\ 5 & 0 & 1 & -1 \\ 2 & -2 & 0 & 3 \end{pmatrix}$ Compute $\alpha$ and $\beta$.

*Solution* For each row, we write down the minimum entry and for each column we write down the maximum entry.

| | | | | | Min |
|---|---|---|---|---|---|
| | 0 | 1 | −3 | 4 | −3 |
| | 5 | 0 | 1 | −1 | −1 |
| | 2 | −2 | 0 | 3 | −2 |
| Max | 5 | 1 | 1 | 4 | |

Thus $\alpha = \max\min a_{ij} = -1$ and $\beta = \min\max a_{ij} = 1$. So $\mathbf{a}_2$ is the conservative strategy for $R$ and either $\mathbf{a}^2$ or $\mathbf{a}^3$ is a conservative strategy for $C$.

**6.7. Definition.** A matrix game is **strictly determined** if $\alpha = \beta$. The common value is the **value** of the game.

There is some interest in the situation if a game is not strictly determined; the pessimistic outcomes for Rowman and Columnman can't both happen.

**6.8. Definition.** An entry $a_{ij}$ in the matrix $\mathbf{A}$ is a **saddle point** if it is smallest in its row and largest in its column.

**6.9. Theorem.** *A matrix game* $\mathbf{A}$ *is strictly determined if and only if* $\mathbf{A}$ *has a saddle point.*

*Proof.* Assume first that $\alpha = \beta$. We show there is an element of $\mathbf{A}$ which is both the smallest in its row, and the largest in its column.

Even without the assumption that $\alpha = \beta$ there are indices $i_0$ and $j_0$ such that

$$\alpha = \max_i \min_j a_{ij} = \min_j a_{i_0 j} \leq a_{i_0 j_0} \leq \max_i a_{i j_0} = \min_j \max_i a_{ij} = \beta.$$

Since we assume $\alpha = \beta$ it follows that each inequality above is an equality. Consider now the element $a_{i_0 j_0}$. Then

$$\alpha = \min_j a_{i_0 j} = a_{i_0 j_0} = \max_i a_{i j_0} = \beta.$$

The first equality only holds if $a_{i_0 j_0}$ is the smallest element in row $i_0$, while the second only holds if it is the largest in column $j_0$.

Conversely, assume that $a_{i_0 j_0}$ is the smallest element in row $i_0$. Then

$$a_{i_0 j_0} = \min_j a_{i_0 j} \leq \max_i \min_j a_{ij} = \alpha.$$

Similarly $a_{i_0 j_0} \geq \beta$, so $\alpha \geq \beta$. In general we have $\alpha \leq \beta$ so in this case, $\alpha = \beta$. □

## 6.3 Mixed Strategies

Games which are not strictly determined can be further analysed by considering what happens when $R$ and $C$ play a game many times. Instead of choosing the same row (or column) on each occasion, $R$ and $C$ can adopt "mixed" strategies and thereby improve their expected payoff.

**6.10. Definition.** A strategy for the player $R$ is a vector $\mathbf{u}$ in $\mathbb{R}^m$ with $\sum_{i=1}^m u_i = 1$ and $\mathbf{u} \geq \mathbf{0}$. A strategy for the player $C$ is a vector $\mathbf{v}$ in $\mathbb{R}^n$ with $\sum_{j=1}^n v_j = 1$ and $\mathbf{v} \geq \mathbf{0}$.

What we have in mind is that if $R$ has strategy $\mathbf{u}$ then he chooses row $\mathbf{a}_i$ with probability $u_i$. That is, over a series of games, he chooses $\mathbf{a}_i$ with the frequency that this specifies. Similarly, if $C$ has strategy $\mathbf{v}$ then he chooses column $\mathbf{a}^j$ with the frequency given by the probability $v_j$.

**6.11. Definition.** For strategies $\mathbf{u}$ for $R$ and $\mathbf{v}$ for $C$, the **expectation** $E(\mathbf{u}, \mathbf{v})$ is given by

$$E(\mathbf{u}, \mathbf{v}) = \sum_{i=1}^{m} \sum_{j=1}^{n} u_i a_{ij} v_j.$$

The expectation gives the 'expected value' of the payoff that $R$ receives from $C$. Using matrix notation, one has $E(\mathbf{u}, \mathbf{v}) = \mathbf{u}^T \mathbf{A} \mathbf{v}$. Note that Rowman is sure to win $\min_{\mathbf{v}} E(\mathbf{u}, \mathbf{v})$ if he uses mixed strategy $\mathbf{u}$. If he now maximises his certain winnings by choosing $\mathbf{u}$ sensibly, he can be sure of winning

$$\alpha^* = \max_{\mathbf{u}} \min_{\mathbf{v}} E(\mathbf{u}, \mathbf{v}).$$

In the same way, Columnman, using strategy $\mathbf{v}$, will lose at most $\max_{\mathbf{u}} E(\mathbf{u}, \mathbf{v})$, and he can reduce this loss as much as possible by choosing his strategy $\mathbf{v}$ sensibly; he then looses

$$\beta^* = \min_{\mathbf{v}} \max_{\mathbf{u}} E(\mathbf{u}, \mathbf{v}).$$

**6.12. Definition.** Let

$$\alpha^* = \max_{\mathbf{u}} \min_{\mathbf{v}} E(\mathbf{u}, \mathbf{v}) \text{ and } \beta^* = \min_{\mathbf{v}} \max_{\mathbf{u}} E(\mathbf{u}, \mathbf{v}).$$

Then $\alpha^*$ and $\beta^*$ are the **optimum lower** and **optimum upper** values of the game $\mathbf{A}$. Any $\mathbf{u}$ such that

$$\min_{\mathbf{v}} E(\mathbf{u}, \mathbf{v}) = \alpha^*$$

is an optimum strategy for $R$ and any $\mathbf{v}$ such that

$$\max_{\mathbf{u}} E(\mathbf{u}, \mathbf{v}) = \beta^*$$

is an optimum strategy for $C$.

## 6.4 The Fundamental Theorem

**6.13. Theorem.** *For any matrix game, $\alpha^* = \beta^*$ and the common value is the **value** of the game. Thus there is an optimum strategy $\mathbf{u}^*$ for $R$ and an optimum strategy $\mathbf{v}^*$ for $C$ such that $E(\mathbf{u}^*, \mathbf{v}^*) = \alpha^* = \beta^*$.*

*Proof.* Note first that the strategy of the game is unaffected by adding the same constant $k$ to each element of the matrix $\mathbf{A}$; in such a case, the payoff to Rowman is simply increased by $k$. So there is no loss of generality in choosing such a $k$ in such a way that $\mathbf{A} > 0$; we assume that this has been done in what follows. Assume now that Rowman uses strategy $\mathbf{u}$, and that Columnman uses the pure strategy $j$, by which we mean that Columnman always chooses column $j$. This combination means that Rowman's winnings are $\sum_{i=1}^{m} a_{ij}u_i$; and the winnings always exceed a threshold $M$, whichever choice Columnman makes if

$$\sum_{i=1}^{m} a_{ij}u_i \geq M \quad \text{for} \quad j = 1, \ldots, n.$$

Thus we have $\mathbf{u}^T\mathbf{A} \geq M\mathbf{1}^T$, with $\mathbf{u} \geq 0$ and $\mathbf{1}^T\mathbf{u} = 1$. Note that by our positivity assumption on $\mathbf{A}$, necessarily $M > 0$. We now rescale, to write $w_i = u_i/M$, so $w_i \geq 0$, $\mathbf{A}^T\mathbf{w} \geq \mathbf{1}$ and $\mathbf{1}^T\mathbf{w} = M^{-1} = z$ (say). We are interested in the maximum guaranteed return, so we need to maximise $M$, and this is clearly the same as minimising $z$. Rowman's problem is then the linear programming problem

$$\text{minimise } \mathbf{1}^T\mathbf{w} \text{ subject to } \mathbf{A}^T\mathbf{w} \geq \mathbf{1} \text{ and } \mathbf{w} \geq \mathbf{0}.$$

Before going any further, we note that this is in the form to be the dual problem of a linear programming problem, namely the problem:

$$\text{maximise } \mathbf{1}^T\mathbf{x} \text{ subject to } A\mathbf{x} \leq \mathbf{1} \text{ and } \mathbf{x} \geq \mathbf{0},$$

in which $\mathbf{c} = \mathbf{1}$ and $\mathbf{b} = \mathbf{1}$, and we note that the first vector is of length $n$ and the second of length $m$.

Now consider the situation from the point of view of Columnman, who wishes to minimise his losses. Assuming that Rowman chooses pure strategy $i$ and that he uses the mixed strategy $\mathbf{v}$, so $\mathbf{v} \geq \mathbf{0}$ and $\mathbf{1}^T\mathbf{v} = 1$, his expectation is to lose $\sum_{j=1}^{m} a_{ij}v_j$. He will lose at most $M'$, whatever choice Rowman makes if

$$\sum_{j=1}^{m} a_{ij}v_j \leq M' \quad \text{for} \quad i = 1, \ldots, m.$$

Clearly Columnman wishes to make $M'$ as small as possible. Again rescaling, we write $x_j = v_j/M'$, and we note, as above that $M' > 0$. The Columnman's problem is to minimise $M'$, or maximise $1/M' = z'$; again we can phrase this as a linear programming problem, specifically, Columnman wishes to

$$\text{maximise } \mathbf{1}^T\mathbf{x} \text{ subject to } A\mathbf{x} \leq \mathbf{1} \text{ and } \mathbf{x} \geq \mathbf{0},$$

which is of course the dual of Rowman's problem.

Using pure strategies, since $M > 0$ and $M' > 0$, we see there is necessarily a feasible solution to each problem; hence by the Duality Theorem, Theorem 5.6, the primal and dual problems both have optimal solutions with the same optimal value, and $\max z' = \min z$, or

$$\beta^* = \min M' = \max M = \alpha^*.$$

□

The value of a matrix game and optimum strategies for the players can be found using the simplex algorithm. The method is to solve Columnman's problem: maximise $\mathbf{1}^T\mathbf{x}$ subject to $\mathbf{A}\mathbf{x} \leq \mathbf{1}$, $\mathbf{x} \geq \mathbf{0}$.

As a first step, one must make $\mathbf{A}$ positive by adding a suitable constant $k$ to each entry of $\mathbf{A}$. Then take an initial tableau as given in Table 6.1 and use the simplex algorithm to

| | $n$ columns | $m$ columns | **e** | **b** |
|---|---|---|---|---|
| | | | 0 | 1 |
| $m$ | | | 0 | 1 |
| rows | $A$ | $I_m$ | $\vdots$ | $\vdots$ |
| | | | 0 | 1 |
| **e** | $-1\ -1\ \ldots\ -1$ | $0\ 0\ \ldots\ 0$ | 1 | 0 |

Table 6.1: Initial tableau for Columnman's problem.

obtain a final tableau show in Table 6.2.

| | $n$ columns | $m$ columns | **e** | |
|---|---|---|---|---|
| | | | 0 | |
| $m$ | | | 0 | |
| rows | | | $\vdots$ | **x** |
| | | | 0 | |
| **e** | $\geq 0 \geq 0\ \ldots\ \geq 0$ | **w** | 1 | $z$ |

Table 6.2: Final tableau for Columnman's problem.

Note that the **w** which appears under the columns corresponding to the slack variables is the optimal solution of the dual problem, as explained in the proof of the Fundamental Theorem of Duality, Theorem 5.6, as shown specifically in Table 5.2.

The value of the game is given by $\alpha^* = \beta^* = 1/z - k$, and optimum strategies for $R$ and $C$ are given by $\mathbf{u}^* = (1/z)\mathbf{w}$ and $\mathbf{v}^* = (1/z)\mathbf{x}$

Notice that $\mathbf{x}$ is not actually the last column of the final tableau as it stands. For $\mathbf{x}$, one must take the whole $n$-vector that gives the solution to the linear programming problem (denoted in Chapter 4 by $\mathbf{x} = (x_1, x_2, \ldots, x_n)$) and not the $m$-vector given by the values conventionally denoted by $u_1, u_2, \ldots, u_m$. This is clarified in the example that follows.

*6.14. Example.* Solve the matrix game given by

$$\mathbf{A} = \begin{pmatrix} -1 & 0 & 2 \\ 4 & 2 & -1 \end{pmatrix}.$$

It is worth checking to see if the game is strictly determined. We have $\alpha = -1$ and $\beta = 2$, so it is not.

Begin by adding $k$ to each entry with say $k = 2$. This ensures there are feasible solutions of the linear programming problem, as described in the proof of the theorem. The initial tableau is given in Table 6.3

| | $\mathbf{a}_1$ | $\mathbf{a}_2$ | $\mathbf{a}_3$ | $\mathbf{a}_4$ | $\mathbf{a}_5$ | $\mathbf{e}$ | $\mathbf{b}$ |
|---|---|---|---|---|---|---|---|
| $\mathbf{a}_4$ | 1 | 2 | 4 | 1 | 0 | 0 | 1 |
| $\mathbf{a}_5$ | 6 | $4^*$ | 1 | 0 | 1 | 0 | 1 |
| $\mathbf{e}$ | $-1$ | $-1$ | $-1$ | 0 | 0 | 1 | 0 |

Table 6.3: Initial tableau for Example 6.14.

We now pivot about the entry indicated. Note that in principle we could have swapped any of $\mathbf{a}_1$, $\mathbf{a}_2$ or $\mathbf{a}_3$ into the basis; we have made the choice more or less at random. Doing the pivot gives the second tableau shown in Table 6.4.

| | $\mathbf{a}_1$ | $\mathbf{a}_2$ | $\mathbf{a}_3$ | $\mathbf{a}_4$ | $\mathbf{a}_5$ | $\mathbf{e}$ | $\mathbf{b}$ |
|---|---|---|---|---|---|---|---|
| $\mathbf{a}_4$ | $-2$ | 0 | $(7/2)^*$ | 1 | $-1/2$ | 0 | $1/2$ |
| $\mathbf{a}_2$ | $3/2$ | 1 | $1/4$ | 0 | $1/4$ | 0 | $1/4$ |
| $\mathbf{e}$ | $1/2$ | 0 | $-3/4$ | 0 | $1/4$ | 1 | $1/4$ |

Table 6.4: Second tableau for Example 6.14.

This time there is only one proper sign for improvement; pivoting as shown, we obtain the final tableau in Table 6.5

| | $\mathbf{a}_1$ | $\mathbf{a}_2$ | $\mathbf{a}_3$ | $\mathbf{a}_4$ | $\mathbf{a}_5$ | $\mathbf{e}$ | $\mathbf{b}$ |
|---|---|---|---|---|---|---|---|
| $\mathbf{a}_3$ | $-4/7$ | 0 | 1 | $2/7$ | $1/7$ | 0 | $1/7$ |
| $\mathbf{a}_2$ | $23/14$ | 1 | 0 | $-1/14$ | $2/7$ | 0 | $3/14$ |
| $\mathbf{e}$ | $1/14$ | 0 | 0 | $3/14$ | $1/7$ | 1 | $5/14$ |

Table 6.5: Final tableau for Example 6.14.

So $z = \frac{5}{14}$, and we can read off the optimal solutions for both the initial problem and the dual; they are

$$\mathbf{w} = \left(\frac{3}{14}, \frac{1}{7}\right) \qquad \text{and} \qquad \mathbf{x} = \left(0, \frac{3}{14}, \frac{1}{7}\right).$$

We thus get the value of the game as

$$\alpha^* = \beta^* = \frac{14}{5} - 2 = \frac{4}{5},$$

where of course we now remove the constant $k$ added on above. The optimum strategy for Columnman is then given by

$$\mathbf{v} = \frac{1}{z}\mathbf{x} = \frac{14}{5}\left(0, \frac{3}{14}, \frac{1}{7}\right) = \left(0, \frac{3}{5}, \frac{2}{5}\right).$$

Here of course we have reported the full solution $\mathbf{x}$, rather than the basic solution corresponding to the last column of the matrix. Similarly the optimal strategy for Rowman is

$$\mathbf{u} = \frac{1}{z}\mathbf{w} = \frac{14}{5}\left(\frac{3}{14}, \frac{1}{7}\right) = \left(\frac{3}{5}, \frac{2}{5}\right).$$

Of course we can work without applying the "shift" by $k$ to the game. Here is the calculation; if anything it is slightly easier than before.

```
> with (linalg):
> A:=matrix(3,7,[-1,0,2,1,0,0,1,4,2,-1,0,1,0,1,-1,-1,-1,0,0,1,0]);
```

$$A := \begin{bmatrix} -1 & 0 & 2 & 1 & 0 & 0 & 1 \\ 4 & 2 & -1 & 0 & 1 & 0 & 1 \\ -1 & -1 & -1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

```
> B:=mulrow(A,1,1/2):B:=pivot(B,1,3);
```

$$B := \begin{bmatrix} \frac{-1}{2} & 0 & 1 & \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ \frac{7}{2} & 2 & 0 & \frac{1}{2} & 1 & 0 & \frac{3}{2} \\ \frac{-3}{2} & -1 & 0 & \frac{1}{2} & 0 & 1 & \frac{1}{2} \end{bmatrix}$$

```
> C:=mulrow(B,2,1/2):C:=pivot(C,2,2);
```

$$C := \begin{bmatrix} \frac{-1}{2} & 0 & 1 & \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ \frac{7}{4} & 1 & 0 & \frac{1}{4} & \frac{1}{2} & 0 & \frac{3}{4} \\ \frac{1}{4} & 0 & 0 & \frac{3}{4} & \frac{1}{2} & 1 & \frac{5}{4} \end{bmatrix}$$

However, we have no guarantee that this calcualtion is relevant to our original problem; indeed if we end up with an objective value which is zero, or even nagtive, it *isn't* useful as you can quickly see if you try to translate back to the original problem. So the original shift by $k$ is important!

*6.15. Example.* Solve the matrix game given by

$$\mathbf{A} = \begin{pmatrix} 4 & 2 \\ 1 & 3 \\ 3 & 4 \end{pmatrix}.$$

*Solution* As before we use the fundamental theorem of Game theory. The full set of tables are given in Table 6.6.

In this example, we have $z = \frac{3}{10}$, and we can read off the optimal solutions for both the initial problem and the dual; they are

$$\mathbf{w} = \left(\frac{1}{10}, 0, \frac{1}{5}\right) \qquad \text{and} \qquad \mathbf{x} = \left(\frac{1}{5}, \frac{1}{10}\right).$$

| | $\mathbf{a}_1$ | $\mathbf{a}_2$ | $\mathbf{a}_3$ | $\mathbf{a}_4$ | $\mathbf{a}_5$ | $\mathbf{e}$ | $\mathbf{b}$ |
|---|---|---|---|---|---|---|---|
| $\mathbf{a}_3$ | $4^*$ | 2 | 1 | 0 | 0 | 0 | 1 |
| $\mathbf{a}_4$ | 1 | 3 | 0 | 1 | 0 | 0 | 1 |
| $\mathbf{a}_5$ | 3 | 4 | 0 | 0 | 1 | 0 | 1 |
| $\mathbf{e}$ | $-1$ | $-1$ | 0 | 0 | 0 | 1 | 0 |
| $\mathbf{a}_1$ | 1 | 1/2 | 1/4 | 0 | 0 | 0 | 1/4 |
| $\mathbf{a}_4$ | 0 | 5/2 | $-1/4$ | 1 | 0 | 0 | 3/4 |
| $\mathbf{a}_5$ | 0 | $5^*/2$ | $-3/4$ | 0 | 1 | 0 | 1/4 |
| $\mathbf{e}$ | 0 | $-1/2$ | 1/4 | 0 | 0 | 1 | 1/4 |
| $\mathbf{a}_1$ | 1 | 0 | 2/5 | 0 | $-1/5$ | 0 | 1/5 |
| $\mathbf{a}_4$ | 0 | 0 | 1/2 | 1 | $-1$ | 0 | 1/2 |
| $\mathbf{a}_2$ | 0 | 1 | $-3/10$ | 0 | 2/5 | 0 | 1/10 |
| $\mathbf{e}$ | 0 | 0 | 1/10 | 0 | 1/5 | 1 | 3/10 |

Table 6.6: Complete tableau for Example 6.15.

The value of the game is $\alpha^* = \beta^* = 10/3$; the optimum strategy for Columnman is then given by

$$\mathbf{v} = \frac{1}{z}\mathbf{x} = \frac{10}{3}\left(\frac{1}{5}, \frac{1}{10}\right) = \left(\frac{2}{3}, \frac{1}{3}\right),$$

while for Rowman it is

$$\mathbf{u} = \frac{1}{z}\mathbf{w} = \frac{10}{3}\left(\frac{1}{10}, 0, \frac{1}{5}\right) = \left(\frac{1}{3}, 0, \frac{2}{3}\right).$$

## 6.5 Questions 5 (Hints and solutions start on page 120.)

*6.1. Q.* Rowman and Columnman play the following game in which they conceal and then display coins. Each first hides in his hand a "stake" of zero, one or two £1 coins. They then reveal what is in their hand at the same time. If both players reveal the same number of coins, Rowman pays to Columnman the total number of coins both have shown; if the numbers of coins are different, Columnman pays Rowman the total number of coins both have shown. Write down the payoff matrix for Rowman and show that the game is not strictly determined.

State the linear programming problem that Columnman has to solve to find his optimal strategy. Assuming that the corresponding final tableau is

| Basis | $\mathbf{a}_1$ | $\mathbf{a}_2$ | $\mathbf{a}_3$ | $\mathbf{a}_4$ | $\mathbf{a}_5$ | $\mathbf{a}_6$ | $\mathbf{e}$ | $\mathbf{b}$ |
|---|---|---|---|---|---|---|---|---|
| $\mathbf{a}_2$ | 0 | 1 | 0 | $\frac{21}{76}$ | $\frac{-9}{38}$ | $\frac{1}{76}$ | 0 | $\frac{1}{19}$ |
| $\mathbf{a}_3$ | 0 | 0 | 1 | $\frac{23}{152}$ | $\frac{1}{76}$ | $\frac{-17}{152}$ | 0 | $\frac{1}{19}$ |
| $\mathbf{a}_1$ | 1 | 0 | 0 | $\frac{-49}{152}$ | $\frac{21}{76}$ | $\frac{23}{152}$ | 0 | $\frac{2}{19}$ |
| $\mathbf{e}$ | 0 | 0 | 0 | $\frac{2}{19}$ | $\frac{1}{19}$ | $\frac{1}{19}$ | 1 | $\frac{4}{19}$ |

determine the optimal strategies for Rowman and Columnman and derive the value of the game. You are *not* asked to derive the final tableau yourself.

*6.2. Q.* The game of "matrix hide and seek" is played between players $H$ and $S$. Player $H$ first chooses an entry in the matrix

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 3 \end{pmatrix}$$

and "hides" at that position. Player $S$ then "seeks" by choosing either one of the two rows or one of the three columns, so has a choice from 5 selections in all. If $H$ is hiding in the row or column selected by $S$ then $H$ pays £$k$ to $S$, where $k$ is the entry in the matrix at which he "hid". Otherwise $S$ pays $H$ £1. Write down the payoff matrix for $S$ and show that the game is not strictly determined.

*6.3. Q.* Suppose that two matrix games are given by

$$\mathbf{A} = \begin{pmatrix} 4 & 2 \\ 8 & 6 \\ 2 & 4 \end{pmatrix} \qquad \mathbf{B} = \begin{pmatrix} 2 & 6 \\ 8 & 4 \\ 4 & 2 \end{pmatrix}$$

where the matrices give the payoff to Rowman. Obtain the upper and lower values $\alpha$ and $\beta$ for each game, and say which game is strictly determined.

Use the simplex method to solve the other game, showing that the value of the game lies between $\alpha$ and $\beta$. Give the optimum strategy for each of the two players.

If you were Rowman, would you, given the choice, play game **A** or game **B**? Why?

*6.4. Q.* Consider the matrix game given by

$$\mathbf{A} = \begin{pmatrix} 3 & 2 \\ 2 & 6 \\ 1 & 5 \end{pmatrix}$$

where the matrix gives the payoff to Rowman. Obtain the upper and lower values $\alpha$ and $\beta$ for this game, and show that the game is not strictly strictly determined. Explain which of Rowman's pure strategies will clearly *not* feature in his optimal strategy.

Express Columnman's problem as a linear programming problem. Write down the initial tableau for its solution and apply the Simplex Algorithm to derive a final tableau.

One way of doing this leads to the final tableau shown in Table 6.7. Using this final tableau state the value of the game, and give optimal strategies for Rowman and Columnman.

| Basis | $\mathbf{a}_1$ | $\mathbf{a}_2$ | $\mathbf{a}_2$ | $\mathbf{a}_4$ | $\mathbf{a}_5$ | $\mathbf{e}$ | $\mathbf{b}$ |
|---|---|---|---|---|---|---|---|
| $\mathbf{a}_1$ | 1 | 0 | 3/7 | −1/7 | 0 | 0 | 2/7 |
| $\mathbf{a}_2$ | 0 | 1 | −1/7 | 3/14 | 0 | 0 | 1/14 |
| $\mathbf{a}_5$ | 0 | 0 | 2/7 | −13/14 | 1 | 0 | 5/14 |
| $\mathbf{e}$ | 0 | 0 | 2/7 | 1/14 | 0 | 1 | 5/14 |

Table 6.7: One possible final tableau for Question 6.4.

# Chapter 7

# Non-linear Optimisation Problems

## 7.1 Relaxing linearity conditions

We have seen in the preceding sections how the assumption of linearity, both of the objective function and the constraints, allowed a very effective optimisation procedure to be developed. In the remainder of these notes, we explore the situation in which these restrictions no longer hold. In other words, we consider the problem

$$\text{maximise } f(\mathbf{x}), \text{ subject to constraints } c_j(\mathbf{x}) = 0 \text{ for } (1 \leq j \leq k).$$

In this generality, nothing useful can be said. The situation is sometimes summed up by saying that both the objective function and the constraints are *non-linear* and is referred to as a non-linear optimisation problem. However the term "non-linear" should not be thought of a descriptive, but simply an indication of what fails. To make the point more directly, it may be quite reasonable to study (say) bananas, but what does it mean to study "non-bananas"? An apple is a non-banana, but so is a University course in optimisation!

Here is an example of what looks like a very simple problem. You are given $n$ distinct points in the plane, say $(x_i, y_i)$ for $1 \leq i \leq n$ and have to minimise the objective function

$$f(x, y) = \sum_{i=1}^{n} \sqrt{(x - x_i)^2 + (y - y_i)^2}$$

Clearly such a minimum exists. You may wish to experiment numerically in the particular case when the three points are at $(0, 0)$, $(1, 0)$ and $(0, 2)$. You should find, a *unique* optimal solution

$$(x*, y*) = (0.3045036, 0.2545693)$$

to seven figures. Although it looks simple, in general there is no explicit solution to this problem; note that in the example above, the optimal solution is surprisingly far away from the centre of mass $(0.333, 0.666)$ of the two points.

This problem is known as the **Fermat-Weber** problem; you can find more information in Kaplan (1999). It is more obviously practical if for example you consider the objective function as the cost of the water pipes which have to be laid in order to connect each house (at position $(x_i, y_i)$) with water from central supply located at $(x, y)$. In this guise, the problem is also known as the **facility location problem**.

In the remainder of this section we take up the study of non-linear optimisation problems in general, but our motivation is to study one of the simplest possible extensions of the linear problem, namely the situation when the objective function, but not the constraints, is allowed to have second order, or quadratic terms as well as linear terms. Then in Chapter 8 we take a very different view, and indicate some modern approaches that can be useful for very general optimisation problems. Finally in Chapter 9 we investigate an intermediate technique which is probabilistic, and so needs modern computing tools, yet builds on much of the traditional technique. A good overview of the variety of methods for such problems can be found in Michalewicz & Fogel (2000). I recommend this quite strongly as an interesting and slightly eccentric book.[1]

It is tempting to say there are *no* general methods, simply a collection of techniques which have been explored for particular types of problems. However there is of course a reliance on simpler techniques we have already studied, and this suggests one approach to more general problems:

- extract one or more linear subproblems; or
- convert the original problem to an unconstrained one, for which simpler methods are available.

Our first step is to note that certain types of constrained optimisation problems are familiar from calculus, and indeed that we traditionally pass to the unconstrained situation.

## 7.2 Maxima and Minima

As in one variable calculations, one use for derivatives in several variables is in calculating maxima and minima. Again as for one variable, we shall rely on the theorem that if $f$ is continuous on a closed bounded subset of $\mathbb{R}^2$, then it has a global maximum and a global minimum. And again as before, we note that these must occur *either* at a local maximum or minimum, or else on the boundary of the region. Of course in $\mathbb{R}$, the boundary of the region usually consisted of a pair of end points, while in $\mathbb{R}^2$, the situation is more complicated. However, the principle remains the same. And we can test for local maxima and minima in the same way as for one variable.

**7.1. Definition.** Say that $f(x, y)$ has a **critical point** at $(a, b)$ if and only if

$$\frac{\partial f}{\partial x}(a, b) = \frac{\partial f}{\partial y}(a, b) = 0.$$

It is clear by comparison with the single variable result, that a necessary condition that $f$ have a local extremum at $(a, b)$ is that it have a critical point there, although that is not a sufficient condition. We refer to this as the **first derivative test**.

We can get more information by looking at the second derivative. Recall that we gave a number of different notations for partial derivatives, and in what follows we use $f_x$ rather than the more cumbersome $\dfrac{\partial f}{\partial x}$ etc. This idea extends to higher derivatives; we shall use

$$f_{xx} \quad \text{instead of} \quad \frac{\partial^2 f}{\partial x^2} \quad \text{and} \quad f_{xy} \quad \text{instead of} \quad \frac{\partial^2 f}{\partial x \partial y} \quad \text{etc.}$$

[1] It has chapter headings such as "Who Owns the Zebra?" and " What's the Color of the Bear?".

**7.2. Theorem (Second Derivative Test).** *Assume that $(a, b)$ is a critical point for $f$. Then*

- *If, at $(a, b)$, we have $f_{xx} < 0$ and $f_{xx}f_{yy} - f_{xy}^2 > 0$, then $f$ has a **local maximum** at $(a, b)$.*
- *If, at $(a, b)$, we have $f_{xx} > 0$ and $f_{xx}f_{yy} - f_{xy}^2 > 0$, then $f$ has a **local minimum** at $(a, b)$.*
- *If, at $(a, b)$, we have $f_{xx}f_{yy} - f_{xy}^2 < 0$, then $f$ has a **saddle point** at $(a, b)$.*

*The test is **inconclusive** at $(a, b)$ if $f_{xx}f_{yy} - f_{xy}^2 = 0$, and the investigation has to be continued some other way.*

Note that the discriminant is easily remembered as

$$\Delta = \begin{vmatrix} f_{xx} & f_{xy} \\ f_{yx} & f_{yy} \end{vmatrix} = f_{xx}f_{yy} - f_{xy}^2.$$

A number of very simple examples can help to remember this. After all, the result of the test should work on things where we can do the calculation anyway!

*7.3. Example.* Show that $f(x, y) = x^2 + y^2$ has a minimum at $(0, 0)$.

Of course we know it has a global minimum there, but here goes with the test:
*Solution* We have $f_x = 2x$; $f_y = 2y$, so $f_x = f_y$ precisely when $x = y = 0$, and this is the only critical point. We have $f_{xx} = f_{yy} = 2$; $f_{xy} = 0$, so $\Delta = f_{xx}f_{yy} - f_{xy}^2 = 4 > 0$ and there is a local minimum at $(0, 0)$.

*7.4. Example.* Let $f(x, y) = xy$. Show there is a unique critical point, which is a saddle point.

*Solution* $f_x = y$; $f_y = x$, and so there is a critical point only at $(0, 0)$. Also $f_{xx} = 0 = f_{yy}$; $f_{xy} = 1$, so $\Delta = -1$ and indeed we have a saddle point at $(0, 0)$.

*Proof.* We give an indication of how the theorem can be derived — or if necessary how it can be remembered. We start with the two dimensional version of Taylor's theorem. We have

$$f(a + h, b + k) \sim f(a, b) + h\frac{\partial f}{\partial x}(a, b) + k\frac{\partial f}{\partial y}(a, b) + \frac{1}{2}\left(h^2\frac{\partial^2 f}{\partial x^2} + 2kh\frac{\partial^2 f}{\partial x \partial y} + k^2\frac{\partial^2 f}{\partial y^2}\right)$$

where we have actually taken an expansion to second order and assumed the corresponding remainder is small.

We are looking at a critical point, so for any pair $(h, k)$, we have $h\dfrac{\partial f}{\partial x}(a, b) + k\dfrac{\partial f}{\partial y}(a, b) = 0$ and everything hinges on the behaviour of the second order terms. It is thus enough to study the behaviour of the quadratic $Ah^2 + 2Bhk + Ck^2$, where we have written

$$A = \frac{\partial^2 f}{\partial x^2} = f_{xx}, \qquad B = \frac{\partial^2 f}{\partial x \partial y} = f_{xy}, \quad \text{and} \quad C = \frac{\partial^2 f}{\partial y^2} = fyy.$$

Assuming that $A \neq 0$ we can write

$$\begin{aligned} Ah^2 + 2Bhk + Ck^2 &= A\left(h + \frac{Bk}{A}\right)^2 + \left(C - \frac{B^2}{A}\right)k^2 \\ &= A\left(h + \frac{Bk}{A}\right)^2 + \left(\frac{\Delta}{A}\right)k^2 \end{aligned}$$

where we write $\Delta = CA - B^2$ for the discriminant. We have thus expressed the quadratic as the sum of two squares. It is thus clear that

- if $A < 0$ and $\Delta > 0$ we have a local maximum;
- if $A > 0$ and $\Delta > 0$ we have a local minimum; and
- if $\Delta < 0$ then the coefficients of the two squared terms have opposite signs, so by going out in two different directions, the quadratic may be made either to increase or to decrease.

Note also that we could have completed the square in the same way, but starting from the $k$ term, rather than the $h$ term; so the result could just as easily be stated in terms of $C$ instead of $A$ □

*7.5. Example.* Find the extrema of $f(x, y) = xy - x^2 - y^2 - 2x - 2y + 4$ in the positive quadrant.

*Solution* Since $f_x = y - 2x - 2$ and $f_y = x - 2y - 2$, we have a critical point when $F - x = f_y = 0$, or $(x, y) = (2, 2)$. Then $f_{xx} = -2$; $f_{yy} = -2$, and $f_{xy} = 1$, and so $\Delta = f_{xx}f_{yy} - f_{xy}^2 = 4 - 1 = 3 > 0$. Since $f_{xx} < 0$, we have a local maximum; in fact it is a global maximum.

Note that, had the objective function been linear, rather than of at most second order, this would have been a very simple linear programming problem with no constraints. We now consider optimisation problems with additional constraints.

*7.6. Example.* An open-topped rectangular tank is to be constructed so that the sum of the height and the perimeter of the base is 30 metres. Find the dimensions which maximise the surface area of the tank. What is the maximum value of the surface area? [You may assume that the maximum exists, and that the corresponding dimensions of the tank are strictly positive.]

*Solution* Let the dimensions of the box be as shown in Fig 7.2, and let the area of the surface of the material be $S$. Then

$$S = 2xh + 2yh + xy,$$

and since, from our restriction on the base and height,

$$30 = 2(x + y) + h, \quad \text{we have} \quad h = 30 - 2(x + y).$$

Substituting, we have

$$S = 2(x + y)\Big(30 - 2(x + y)\Big) + xy = 60(x + y) - 4(x + y)^2 + xy,$$

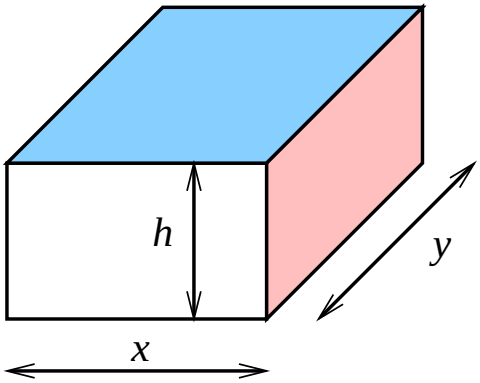


Figure 7.1: A dimensioned box

and for physical reasons, $S$ is defined for $x \geq 0$, $y \geq 0$ and $x + y \leq 15$.

A global maximum (which we are given exists) can only occur on the boundary of the domain of definition of $S$, or at a critical point, when $\dfrac{\partial S}{\partial x} = \dfrac{\partial S}{\partial y} = 0$. On the boundary of the domain of definition of $S$, we have $x = 0$ or $y = 0$ or $x + y = 15$, in which case $h = 0$. We are given that we may ignore these cases. Now

$$\begin{aligned} S &= -4x^2 - 4y^2 - 7xy + 60x + 60y, \quad \text{so} \\ \frac{\partial S}{\partial x} &= -8x - 7y + 60 = 0, \\ \frac{\partial S}{\partial y} &= -8y - 7x + 60 = 0. \end{aligned}$$

Subtracting gives $x = y$ and so $15x = 60$, or $x = y = 4$. Thus $h = 14$ and the surface area is $S = 16(-4 - 4 - 7 + 15 + 15) = 240$ square metres. Since we are given that a maximum exists, this must be it. [If both sides of the surface are counted, the area is doubled, but the critical proportions are still the same.]

## 7.3 Lagrange's Principle

We saw that one way to solve a *constrained* optimisation problem was to use the constraints to eliminate some of the variables, and so reduce to an unconstrained problem using fewer variables. This is effective when it works, but although the elimination is almost always possible in principle,[2] in practice we may not be able to get the explicit formula need to carry on with an analytic solution. A more organised way of solving such problems uses *Lagrange's* principle.

**7.7. Theorem (Lagrange's Principle).** *Suppose there are constants* $\lambda_1, \lambda_2, \ldots, \lambda_k \in \mathbb{R}$ *such that* $\mathbf{x}_0 \in \mathbb{R}^n$ *is an* unconstrained *maximum of*

$$f(\mathbf{x}) - \sum_{j=1}^{k} \lambda_j c_j(\mathbf{x})$$

[2]It can be done iff the conditions of the implicit function theorem hold.

*and that in addition* $c_j(\mathbf{x}_0) = 0$ *for* $1 \leq j \leq k$. *Then* $\mathbf{x}_0$ *is a solution of the* constrained *maximisation problem:*

$$\textit{maximise } f(\mathbf{x}) \textit{ subject to } c_j(\mathbf{x}) = 0 \textit{ for } 1 \leq j \leq k.$$

*Proof.* Suppose we have such a local maximum $\mathbf{x}_0$. Then there is a neighbourhood $U$ of $\mathbf{x}_0$ such that

$$f(\mathbf{x}_0) = f(\mathbf{x}_0) - \sum_{j=1}^{k} \lambda_j c_j(\mathbf{x}_0) \geq f(\mathbf{x}) - \sum_{j=1}^{k} \lambda_j c_j(\mathbf{x}) \qquad (\mathbf{x} \in U).$$

If we now restrict attention to those points in $U$ for which $c_j(\mathbf{x}) = 0$ for $(1 \leq j \leq k)$, we have $\sum_{j=1}^{k} \lambda_j c_j(\mathbf{x}) = 0$, and so

$$f(\mathbf{x}_0) \geq f(\mathbf{x}) \qquad (\mathbf{x} \in U \quad \text{and} \quad c_j(\mathbf{x}) = 0 \quad \text{for} \quad 1 \leq j \leq k).$$

Thus $\mathbf{x}_0$ is a constrained maximiser of $f$. □

*7.8. Remark.* The same argument shows that if there is a global maximum of the unconstrained problem which satisfies the same conditions, this will gave a global constrained maximum. Replacing $f$ by $-f$ gives the same results for minima.

In practice it is relatively easy to apply this principle, provided we have enough differentiability to allow calculus-based optimisation. Let

$$L(\mathbf{x}) = L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) - \sum_{j=1}^{k} \lambda_j c_j(\mathbf{x})$$

and say that $L$ is the **Lagrangian function**. We seek values $\mathbf{x}_0 \in \mathbb{R}^n$ and $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_k)$ such that $L(\mathbf{x})$ has a local maximum and in addition $c_j(\mathbf{x}_0) = 0$ for $1 \leq j \leq k$. One way to do this is to look for critical points — those points at which

$$\frac{\partial L}{\partial x_i} = 0 \qquad (1 \leq i \leq n).$$

Note also that

$$\frac{\partial L}{\partial \lambda_j} = c_j(\mathbf{x}) \quad (1 \leq j \leq k) \quad \text{and so} \quad c_j(\mathbf{x}) = 0 \quad (1 \leq j \leq k) \quad \text{iff} \quad \frac{\partial L}{\partial \lambda_j} = 0 \quad (1 \leq j \leq k).$$

Thus if we require that $L(\mathbf{x}, \boldsymbol{\lambda})$ has a critical point, as a function of both $\mathbf{x}$ and $\boldsymbol{\lambda}$, or equivalently that

$$\nabla L(\mathbf{x}, \boldsymbol{\lambda}) = 0,$$

we have a point at which we can potentially apply Lagrange's principle. Formally we still have more checking to do, but usually the hard part is to find the point in the first place.

*7.9. Example.* Use Lagrange's principle to re-work Example 7.6.

*Solution* We use Lagrange's method to derive a potential critical point. We are required to maximise $S = 2xh + 2yh + xy$ subject to the constraint that $h + 2(x + y) = 30$. Let

$$L(\mathbf{x}, \lambda) = 2xh + 2yh + xy - \lambda(h + 2x + 2y - 30).$$

Then at a critical point of $L(\mathbf{x}, \lambda)$ we have

$$\frac{\partial L}{\partial x} = 2h + y - 2\lambda = 0, \qquad \frac{\partial L}{\partial y} = 2h + x - 2\lambda = 0,$$
$$\frac{\partial L}{\partial h} = 2x + 2y - \lambda = 0, \qquad \frac{\partial L}{\partial \lambda} = h + 2x + 2y - 30.$$

The third equation gives $\lambda = 2(x + y)$, while from the first two, $y - x = 0$, so $x = y$. Thus $\lambda = 4x$ and $2h = 7x$. Using the constraint we see that $7x/2 + 4x = 30$, so $x = 4$. Thus there is a single critical point of $L$, which occurs at $x = y = 4$ and $h = 14$. Finally we note that this solution satisfies the (implicit) reality constraints, that $x \geq 0$, $y \geq 0$ and $h \geq 0$.

*7.10. Example.* Assume now that the open topped box of Example 7.6 is to have a fixed surface area. Find the proportions so that the volume be a maximum.

*Solution* We are required to maximise $V = xyh$ subject to the constraint that the surface area $2h(x + y) + xy$ is some constant, say $K$. Let

$$L(\mathbf{x}, \lambda) = xyh - \lambda(2h(x + y) + xy - K).$$

Then at a critical point of $L(\mathbf{x}, \lambda)$ we have

$$\frac{\partial L}{\partial x} = yh - 2h\lambda - y\lambda = 0, \qquad \frac{\partial L}{\partial y} = xh - 2h\lambda - x\lambda = 0,$$
$$\frac{\partial L}{\partial h} = xy - 2\lambda(x + y) = 0, \qquad \frac{\partial L}{\partial \lambda} = 2h(x + y) + xy - K = 0.$$

The first two equations give, $(y - x)h = \lambda(y - x)$, so $x = y$ or $h = \lambda$. If $h = \lambda$, the first equation shows that $h\lambda = 0$. We exclude $h = 0$ on physical grounds, because we know the maximum volume will be strictly positive. Thus if $h\lambda = 0$ we must have $\lambda = 0$ and the first equation then shows that $hy = 0$, which we exclude for the same reason. Thus we must have $x = y$.

The third equation now gives $\lambda = x/4$; we then solve the first equation to see that $h = y/2$. Thus there is a local constrained maximum in the volume when the box is square with height half that of the length of the side.

## 7.4 Inequality Constraints

We now consider the more general situation in which the constraints are allowed to be inequalities rather than equalities. Of course we could use the usual "slack variables' technique to stick with "equality" constraints, but there are advantages in working with the minimum number of variables. We thus consider problems of the form:

$$\text{minimise } f(\mathbf{x}) \text{ subject to } c_j(\mathbf{x}) \geq 0 \text{ for } 1 \leq j \leq k. \text{ (NLCO)}$$

A typical example might be the pair

$$c_1(\mathbf{x}) = 1 - x_1^2 - x_2^2, \qquad c_2(\mathbf{x}) = x_2 - x_1^2.$$

The points satisfying this constraint lie within the unit sphere and above the parabola $x_2 = x_1^2$. It is clear how we can minimise an affine function geometrically subject to these constraints.

**7.11. Definition.** We say the $j^{\text{th}}$ constraint is *active* or *tight* at $\mathbf{x}$ if $c_j(\mathbf{x}) = 0$. Otherwise we say the constraint is slack.

Let $\mathbf{x}^*$ be a solution of the constrained non-linear optimisation problem (NLCO). We find it convenient to describe the set of tight constraints, so let $I = \{1, 2, \ldots, k\}$ and let

$$I^* = \{j \in I \mid c_j(\mathbf{x}^*) = 0\}$$

so that $I^*$ is the set of indices of tight constraints. Thus $\mathbf{x}*$ minimises $f(\mathbf{x})$ with $c_j(\mathbf{x}^*) = 0$ for $j \in \mathbf{I}^*$.

We now ignore for the moment the fact that, until we have found $\mathbf{x}^*$ we know nothing about the set $I^*$; in that case, we are trying to solve a constrained maximisation principle with equality constraints, and so can use Lagrange's principle, via the Lagrangian, and seek $\mathbf{x}^*$ and $\lambda_j \quad (j \in I^*)$ at points where

$$\nabla L(\mathbf{x}, \boldsymbol{\lambda}) = \nabla \left( f - \sum_{j \in I^*} \lambda_j c_j(\mathbf{x}) \right) = 0$$

Now define $\lambda_j = 0$ for $j \in I \backslash I^*$. This means that the "full Lagrangian" $f - \sum_{j \in I} \lambda_j c_j$ is defined, and we have

$$\nabla L(\mathbf{x}, \boldsymbol{\lambda}) = \nabla \left( f(\mathbf{x}) - \sum_{j \in I} \lambda_j c_j(\mathbf{x}) \right) (\mathbf{x}^*) = 0$$

since the extra terms are identically zero. We thus have $\lambda_j \mathbf{c}_j(\mathbf{x}^*) = 0$ for all $j$, since either $\lambda_j = 0$ for slack constraints, or $\mathbf{c}_j(\mathbf{x}^*) = 0$ for active constraints.

We now consider the sign of $\lambda_j$, giving an heuristic justification for the fact that $\lambda_j \geq 0$ for all $j$. Recall that for any function $g(\mathbf{x})$, the derivative $\nabla g$ is a vector which points in the direction of increasing values of $g$. Thus if $c_j$ is a tight constraint, and $\mathbf{x}^*$ is the minimiser, $\nabla c_j(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^*}$ points in the direction of increasing $c_j$ and hence *into* the feasible region. Assume now that for the minimiser $\mathbf{x}^*$ the *only* tight constraint is $c_j$. Since $\nabla L(\mathbf{x}, \boldsymbol{\lambda}) = 0$, we have

$$\nabla f(\mathbf{x}^*) = \sum_{j=1}^{n} \lambda_j c_j(\mathbf{x}^*) = \lambda_j \nabla c_j(\mathbf{x}^*).$$

If $\lambda_j < 0$, $\nabla f(\mathbf{x}^*)$ points out of the feasible region, and hence we can reduce $f$ by moving *into* the feasible region. This contradicts the fact that $\mathbf{x}^*$ is a local minimiser for $f$.

The argument is more complicated if two or more constraints are tight at the same time, but this gives the flavour. What we have established is that at $\mathbf{x}^*$ the **Kuhn-Tucker** conditions hold.

**7.12. Definition (Kuhn-Tucker Conditions).** The following three sets of conditions make up the Kuhn-Tucker conditions:

- $\nabla \left( f - \sum_{j=1}^{k} \lambda_j c_j \right) (\mathbf{x}) = 0;$
- $c_j(\mathbf{x}) \geq 0$ and $\lambda_j \geq 0 \quad (1 \leq j \leq k)$; and
- $\lambda_j c(\mathbf{x}) = 0 \quad (1 \leq j \leq k)$.

We have intentionally not been precise about the differentiability conditions needed to get these results. Formally we have the following theorem

**7.13. Theorem (First Kuhn - Tucker).** *Let $\mathbf{x}^*$ be a local minimiser for the problem*

$$\text{minimise } f(\mathbf{x}) \text{ subject to } c_j(\mathbf{x}) \geq 0, \quad (1 \leq j \leq k).$$

*Then provided the objective function $f$ and the constraints $c_j$ are (twice) continuously differentiable, and the set*

$$\{\nabla c_j(\mathbf{x}^*) \mid j \in I^*\}$$

*in linearly independent in $\mathbb{R}^n$, there are constants $\lambda_1, \lambda_2, \ldots, \lambda_k$ such that the Kuhn-Tucker conditions are satisfied.*

*7.14. Remark.* The derivative condition is almost always satisfied, only being violated when two constraints happen to be tangential.

*7.15. Example.* Minimise $x^2 + 2y^2$ subject to $x + y \geq 2$ and $x \leq y$.

*Solution* At this stage, we simply seek possible local minimisers. We have

$$L(x, y, \lambda, \mu) = x^2 + 2y^2 - \lambda(x + y - 2) - \mu(y - x).$$

where we have arranged the constraints and the signs of $\lambda$ and $\mu$ to be consistent with our definition of the "full Lagrangian". The Kuhn - Tucker conditions thus become:

$$\frac{\partial L}{\partial x} = 2x - \lambda + \mu = 0, \qquad \frac{\partial L}{\partial y} = 4y - \lambda - \mu = 0, \tag{7.1}$$

$$x + y \geq 2, \qquad y - x \geq 0, \tag{7.2}$$

$$\lambda \geq 0, \qquad \mu \geq 0, \tag{7.3}$$

$$\lambda(x + y - 2) = 0, \qquad \mu(y - x) = 0. \tag{7.4}$$

We now solve this set of constraints. If $\lambda = 0$ we have $2x + \mu = 0$ and $4y = \mu$. Thus $x + y = -\mu/2 + \mu/4 = -\mu/4$. Since we know that $\mu \geq 0$, we see that $x + y \leq 0$ and so cannot satisfy the constraint $x + y \geq 2$. It follows that $\lambda \neq 0$.

Next consider what happens if we assume that $\mu = 0$. Then arguing as before, we get $2x = \lambda$ and $4y = \lambda$ so that $x = \lambda/2$ and $y = \lambda/4$. Since $\mu \geq 0$, we cannot satisfy the constraint $y - x \geq 0$. It follows that $\mu \neq 0$.

We have now shown that each constraint is tight, so we are, in effect solving the simpler problem in which the constraints are equalities. We have $x = y$, and then $x = y = 1$, which is in fact a global minimiser.

*7.16. Example.* How does the above example change when the second constraint becomes $ay \geq x$ for some $a \geq 2$?

*Solution* The example is of course very similar, until we come to examine whether the constraints are tight or not. We have

$$L(x, y, \lambda, \mu) = x^2 + 2y^2 - \lambda(x + y - 2) - \mu(ay - x).$$

where we have arranged the constraints and the signs of $\lambda$ and $\mu$ to be consistent with our definition of the "full Lagrangian". The Kuhn - Tucker conditions thus become:

$$\frac{\partial L}{\partial x} = 2x - \lambda + \mu = 0, \qquad \frac{\partial L}{\partial y} = 4y - \lambda - a\mu = 0, \tag{7.5}$$

$$x + y \geq 2, \qquad ay - x \geq 0, \tag{7.6}$$

$$\lambda \geq 0, \qquad \mu \geq 0, \tag{7.7}$$

$$\lambda(x + y - 2) = 0, \qquad \mu(ay - x) = 0. \tag{7.8}$$

We now solve this set of constraints. If we assume that $\mu = 0$, we get $2x = \lambda$ and $4y = \lambda$ so that $x = \lambda/2$ and $y = \lambda/4$. Since $x + y \geq 2$, we cannot have $\lambda = 0$, so must have $x + y = 2$, and thus $\lambda = 8/3$, $x = 4/3$ and $y = 2/3$. Since $a \geq 2$ this satisfies the second constraint, and the corresponding value of the objective function is $32/9$.

If $\lambda = 0$ we have $2x + \mu = 0$ and $4y = a\mu$. In fact this gives a valid solution if $a > 2$, but we can rule it out immediately, since we have $x \leq 0$. Since $x + y = 2$, we have $y \geq 2$ and the objective value is at least 4, and so not a minimum value.

## 7.5 Convexity

We conclude this section by giving a geometrical condition which enables us to guarantee solutions are available using the above techniques. One way to guarantee good behaviour is to work with *convex* functions. A convex set is geometrically very simple. A set $C$ is **convex** if $\mathbf{x}^1 \in C$, $\mathbf{x}^2 \in C$ means that $\lambda\mathbf{x}^1 + (1 - \lambda)\mathbf{x}^2 \in C$ whenever $0 \leq \lambda \leq 1$. In other words, if any line joining two points in $c$ lies entirely in $C$.

You may already have met a convex function — one in which the points above the graph of the function (the "supergraph") form a convex set. Equivalently, a function is convex if the line joining two points on the graph of $f$ always lies above the graph of $f$.

An obvious way to generate a convex set is as the intersection of a collection of half planes; hence the set of feasible solutions of a linear programming problem forms a (possibly empty) convex set.

There are other very natural ways to generate convex sets. Let $\mathbf{x}^1$, $\mathbf{x}^2$, ... , $\mathbf{x}^k$ be elements of $\mathbf{R}^n$. It is essentially trivial to show that the set of all convex combinations of $\mathbf{x}^1$, $\mathbf{x}^2$, ... , $\mathbf{x}^k$ is a convex set. Here is the argument: let $\mathbf{u} = \lambda_1\mathbf{x}^1 + \lambda_2\mathbf{x}^2 + \ldots + \lambda_k\mathbf{x}^k$ and $\mathbf{v} = \mu_1\mathbf{x}^1 + \mu_2\mathbf{x}^2 + \ldots + \mu_k\mathbf{x}^k$ be two convex combinations of $\mathbf{x}^1$, $\mathbf{x}^2$, ... , $\mathbf{x}^k$. Write down $\lambda\mathbf{u} + (1 - \lambda)\mathbf{v}$ and show that, when $0 \leq \lambda \leq 1$, this is also a convex combination of $\mathbf{x}^1$, $\mathbf{x}^2$, ... , $\mathbf{x}^k$.

It is of interest that the set of optimal solutions of a linear programming problem is a convex set. The next two examples discuss this:

*7.17. Example.* Show that if $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^k$ are optimal solutions of a linear programming problem, then any convex combination of $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^k$ is also an optimal solution.

*Solution* Let $\lambda_1\mathbf{x}^1 + \lambda_2\mathbf{x}^2 + \ldots + \lambda_k\mathbf{x}^k$ be any convex combination of $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^k$. Let the objective function be $c_1x_1 + c_2x_2 + \ldots + c_nx_n$. Since $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^k$ all have the same objective value, $z$ say, then, for all $i$,

$$c_1x_1^i + c_2x_2^i + \ldots + c_nx_n^i = z,$$

where $\mathbf{x}^i = (x_1^i, x_2^i, \ldots, x_n^i)$. So (why ?),

$$c_1(\lambda_1x_1^1 + \ldots + \lambda_kx_1^k) + \ldots + c_n(\lambda_1x_n^1 + \ldots + \lambda_kx_n^k) = z,$$

which shows (why ?) that $\lambda_1\mathbf{x}^1 + \lambda_2\mathbf{x}^2 + \ldots + \lambda_k\mathbf{x}^k$ also has objective value $z$. Hence the result.

This solution can be written more simply by using matrix notation, letting the objective function be $\mathbf{c}^T\mathbf{x}$.

*7.18. Example.* Prove (by induction on $k$) that, if $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^k$ belong to a convex set $C$, then any convex combination of $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^k$ belongs to $C$.

*Solution* Let $\mathbf{v} = \lambda_1\mathbf{x}^1 + \lambda_2\mathbf{x}^2 + \ldots + \lambda_k\mathbf{x}^k$ be any convex combination of $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^k$. Write $\mathbf{v} = (1 - \lambda_k)\mathbf{u} + \lambda_k\mathbf{x}^k$, where

$$\mathbf{u} = \frac{\lambda_1}{1-\lambda_k}\mathbf{x}^1 + \frac{\lambda_2}{1-\lambda_k}\mathbf{x}^2 + \ldots + \frac{\lambda_{k-1}}{1-\lambda_k}\mathbf{x}^{k-1}.$$

By the induction hypothesis (why ?), $\mathbf{u}$ belongs to $C$ and hence $\mathbf{v}$ belongs to $C$.

We now return to our main thread; that convexity gives us a handle on uniqueness. It is geometrically clear that a (strictly) convex function cannot have two different local minima, while if the same value occurs as a local minimum at different places, the graph must be flat between them, so neither minimum is isolated (and $f$ can't be strictly convex).

Say that $f$ is **concave** if $-f$ is convex. To help remember the difference, note that $x^2$ is convex, while $-x^2$ is concave.

Taylors theorem in its general form allows us to analyse $f$. At the point $\mathbf{x}$, we have

$$f(\mathbf{x}+\mathbf{h}) = f(\mathbf{x}) + \mathbf{h}.\nabla f(\mathbf{x}) + \frac{1}{2}\mathbf{h}^T G(\mathbf{x})\mathbf{h} + \text{higher order terms}$$

exactly as in the two variable case. We have written

$$G = \begin{pmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1 \partial x_n} \\ \dfrac{\partial^2 f}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & \cdots & \dfrac{\partial^2 f}{\partial x_2 \partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \dfrac{\partial^2 f}{\partial x_n \partial x_1} & \dfrac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_n^2} \end{pmatrix}$$

With our usual assumptions, $G(\mathbf{x})$ is symmetric and so it can be diagonalised; let $\lambda_1$, $\lambda_2, \ldots, \lambda_n$ be the eigenvalues of $G(\mathbf{x})$ and write $D(\mathbf{x}) = D = diag(\lambda_1, \lambda_2, \ldots, \lambda_n)$. Then there is an orthogonal $P$ (ie $P^T = P^{-1}$ such that $G(\mathbf{x}) = P^TDP$, and we have

$$f(\mathbf{x}+\mathbf{h}) = f(\mathbf{x}) + \mathbf{h}.\nabla f(\mathbf{x}) + \frac{1}{2}\mathbf{h}^T P^T D(\mathbf{x})(P\mathbf{h}) + \text{higher order terms}$$

At a critical point, $\nabla f(\mathbf{x}) = 0$ and so

$$f(\mathbf{x}+\mathbf{h}) = f(\mathbf{x}) + \frac{1}{2}\left(\lambda_1 u_1^2 + \lambda_2 u_2^2 + \cdots + \lambda_n u_n^2\right) + \text{higher order terms}$$

where $P\mathbf{h} = \mathbf{u}$. Thus $f$ is convex iff $\lambda_i \geq 0$ for all $i$; we can read the behaviour of $f$ from the set of eigenvalues of the Hessian. This leads to:

**7.19. Theorem (Second Derivative Theorem).** *Assume that $f$ is twice differentiable on the convex set $D$. Then $f$ is convex on $D$ if its Hessian $G$ is positive semidefinite, so $\mathbf{x}^T G\mathbf{x} \geq 0$ for each $\mathbf{x}$.*

Note that if $f$ is linear then it is both convex and concave. The function $f(x) = x_1^2 + \cdots + x_n^2$ is strictly convex, since

$$G = \begin{pmatrix} 2 & 0 & \ldots & 0 \\ 0 & 2 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 2 \end{pmatrix}$$

**7.20. Theorem (Second Kuhn-Tucker).** *Let $f : D \to \mathbb{R}^n$ be a* convex *function and suppose that each constraint $c_j$ is* concave *on $D$. Assume that each of the functions involved is twice continuously differentiable. Then every solution of the Kuhn-Tucker conditions is a global minimum of the corresponding NLCO problem.*

## 7.6 Questions 6 (Hints and solutions start on page 124.)

*7.1. Q.* Describe Lagrange's principle briefly, and illustrate your answer by obtaining the maximum value of $x_1^2 + \cdots + x_n^2$ subject to the constraint that $x_1 + \cdots + x_n = 1$.

*7.2. Q.* a) Solve the optimisation problem:

$$\text{minimise } x_1^2 + x_2^2 + x_3^2 \text{ subject to}$$

$$\begin{aligned} x_1 + 2x_2 + \ x_3 &= 4, \\ x_1 - 2x_2 + 2x_3 &= 17. \end{aligned}$$

b) Write down the Kuhn-Tucker conditions for the problem:

$$\text{minimise } x_1^2 + x_2^2 + x_3^2, \text{ subject to}$$

$$\begin{aligned} x_1 + 2x_2 + \ x_3 &\geq 4, \\ x_1 - 2x_2 + 2x_3 &\leq 17. \end{aligned}$$

Find the global minimiser.

*7.3. Q.* The function $z = x_1 - x_2$ is to be minimised subject to the constraints that

$$\begin{aligned} x_1^2 + x_2^2 &\leq 1, \\ 2x_1^2 - 3x_2 &\leq 0. \end{aligned}$$

Sketch the feasible region. Hence obtain geometrically the optimal value and the corresponding values of $x_1$ and $x_2$.

Write down the Kuhn-Tucker conditions for this problem. Illustrate their use by deriving the solution you have just obtained. What information do you get about the tightness of the constraints?

# Chapter 8

# Genetic Optimisation

In Section 7 we discussed optimisation methods based on calculus techniques, showing that although effective, the methods were based on the assumption that problems were quite close to being linear. In contrast, in this section we describe a modern computer - based method which makes no use of calculus. There are obvious reasons why calculus-based methods may be inadequate:

- the function may naturally have sharp corners;
- the function may be ill conditioned;
- the function may have a very large number of critical points; or
- the function may simply be given numerically, rather than with an explicit formula, so all differentiation has to be done that way as well.

There is no single method to be described here, but rather a family of methods, called *genetic algorithms* which seek to evolve solutions to optimisation problems using similar sorts of "survival of the fittest" methods which occur in nature. Further details and many references can be found in Goldberg (1989), which gives a relatively easy introduction to the subject, or the more recent Mitchell (1996). A slightly less discursive treatment, with more emphasis on mathematical optimisation problems is in Michalewicz (1996)

## 8.1 A Simple Algorithm

We begin this study by applying the method to a very simple problem; that of maximising the function $x^2$ over a closed interval. The choice of such a trivial problem means we can concentrate on the methodology, while being able to see how the algorithm has progressed towards the optimum at any time.

### 8.1.1 Coding the Domain

We shall be concerned both with the values of an integer $k$ and also with its representation. Note that 15 and 16 are adjacent values, but have very different (binary) representations as is clear from the sample in Table 8.1. This difference is sometimes called a "Hamming cliff" to describe the sharp discontinuity.

In much of what we do, this can cause problems. However, there is a simple solution, known as **Gray coding** which avoids the problem, by ensuring that adjacent numbers have "binary" codes which differ by at most one bit.[1] If in addition, we require Gray coded integers to have the same code length as their binary values, and that variation occurs from the "bottom up" when there is a choice, this is enough to define the code by induction. Table 8.2 gives the first few values.

| $n$ | Code | $n$ | Code |
|---|---|---|---|
| 0 | 00000 | 8 | 01000 |
| 1 | 00001 | 9 | 01001 |
| 2 | 00010 | 10 | 01010 |
| 3 | 00011 | 11 | 01011 |
| 4 | 00100 | 12 | 01100 |
| 5 | 00101 | 13 | 01101 |
| 6 | 00110 | 14 | 01110 |
| 7 | 00111 | 15 | 01111 |

Table 8.1: Binary codes.

| $n$ | Code | $n$ | Code |
|---|---|---|---|
| 0 | 00000 | 8 | 01100 |
| 1 | 00001 | 9 | 01101 |
| 2 | 00011 | 10 | 01111 |
| 3 | 00010 | 11 | 01110 |
| 4 | 00110 | 12 | 01010 |
| 5 | 00111 | 13 | 01011 |
| 6 | 00101 | 14 | 01001 |
| 7 | 00100 | 15 | 01000 |

Table 8.2: Gray codes.

We can now state the form of optimisation problem we shall deal with. We assume given a function $f = f(k_1, \ldots, k_p)$ of $p$ parameters or variables, each of which has a finite set of possible values. We assume in addition that $f$ is normalised, perhaps by adding a suitable constant, so that $f(k_1, \ldots, k_p) \geq 0$. Our problem is then:

maximise $f$ subject to these restrictions. (GA)

The restriction to a finite number of parameters, each of which can take a finite set of values means we can encode a possible solution of such a problem as a *string* of bits, of length $n$, say. For convenience we assume conversely that all such strings do give valid members of the domain of $f$.

This is a considerable simplification, since at least in this case we are restricting to an *unconstrained* maximisation problem where standard techniques such as gradient ascent are available. We have already seen that constraints seem almost inevitable in real applications yet in general increase the difficulty in obtaining an optimum value very significantly. We show in Section 8.4 that a genetic algorithm can incorporate constraints with less difficulty than some other methods.

With this assumption, the set of all possible solutions thus has size $2^n$, and a typical solution is then

$$\mathbf{s} = (0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, \ldots, 1, 0) \in \{0, 1\}^n.$$

Note that this string is an encoding of the particular choice of parameter values. We make no assumptions about what type of parameters they are, or how the encoding is done. If each parameter is a number, one way to code is first to create suitable binary (or Gray) codes for each parameter, and then concatenate the resulting strings to get a single string.

[1] Recall that the word "bit" is an abbreviation for "binary digit"; the word "digit" alone implies a scale of ten.

We then write $f(\mathbf{s})$ for the value $f(k_1, \ldots, k_p) \geq 0$, where $\mathbf{s}$ is the string which is the encoding of $(k_1, \ldots, k_p)$, and call $f(\mathbf{s})$ the *fitness* of the string $\mathbf{s}$. It is natural then to say that $\mathbf{s}_1$ is *fitter* than $\mathbf{s}_2$ if $f(\mathbf{s}_1) \geq f(\mathbf{s}_2)$. Thus our problem becomes that of finding the fittest strings from our string universe $\{0,1\}^n$.

*8.1. Example.* A black box is fitted with five switches, each of which can be either "on" or "off". The output of the box (perhaps electrical potential or "voltage") can be measured, and varies according to the position of the switches. Find the positions of the switches which maximise the output of the box.

*Solution* We develop a full solution slowly, but note there is an obvious binary encoding of the solution space as follows:

$$(0,0,1,1,0) \longleftrightarrow (\text{off}, \text{off}, \text{on}, \text{on}, \text{off}).$$

Our main example is a simple mathematical one which presents no difficulties. Let $f(x) = 1024x^2$. We wish to maximise $f$ on the domain $[0,1)$, and work in units of $1/32$, so in practice we seek the maximum of the finite set

$$f(0), f\left(\frac{1}{32}\right), f\left(\frac{2}{32}\right), \ldots, f\left(\frac{31}{32}\right).$$

Of course $f$ is monotone, so the maximum is $31^2 = 961$; we derive thus result using a genetic algorithm.

### 8.1.2 Genetic algorithms: process model

We first describe the model which motivates the detailed description of a genetic algorithm. In the model:

- each string is a gene;
- natural selection favours fit strings;
- strings can be combined to produce new strings;
- there is a rare random process called mutation which alters strings locally.

We illustrate the process using our example and pass through a number of stages

**Gene Pool** We first need to choose a population of strings, the *gene* pool, from which the solution is to evolve. The assumption here is that it is not practicable to allow a gene pool consisting of the universe $\{0,1\}^n$, otherwise the method simply reduces to that of "exhaustive search". For illustration, we choose a gene pool with population size $P = 4$ strings; in general the size of the gene pool is one of a number of choices that has to be made by the designer of the algorithm. Our gene pool in this case is chosen "at random", and the fitness of each string or gene is evaluated to give Table 8.3.

| **s** | $32x$ | $f(x)$ |
|---|---|---|
| 01101 | 13 | 169 |
| 11000 | 24 | 576 |
| 01000 | 8 | 64 |
| 10011 | 19 | 361 |

Table 8.3: Initial Gene Pool.

**Natural Selection** We now allow the fittest strings to survive and enter the mating pool to produce a new generation. There are many ways of doing this, and at the same time keeping the population size $P$ constant. One natural way is to select is to choose strings at random with a probability proportional to the fitness of the string. In general, this will mean that some strings are selected more than once, while others, usually the least fit, drop out of the gene pool. In our example we assume that such a selection process included the fittest string twice, and allowed the least fit string to disappear, giving the *mating* pool of Table 8.4

| **s** | $32x$ | $f(x)$ |
|---|---|---|
| 01101 | 13 | 169 |
| 11000 | 24 | 576 |
| 11000 | 24 | 576 |
| 10011 | 19 | 361 |

Table 8.4: Mating Pool.

**Mating** We assume for simplicity that each pair of parent strings produce exactly two offspring.[2] The genes of the offspring are obtained by mixing parental genes in an operation know as **crossover**. This is illustrated in Fig. 8.1 The number of cross-overs, and the sites of each cross-over are chosen at random; in the example, there are two cross-over sites. The offspring are shown in Fig. 8.2.

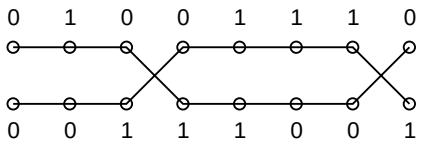


Figure 8.1: Two strings with two cross-over sites.

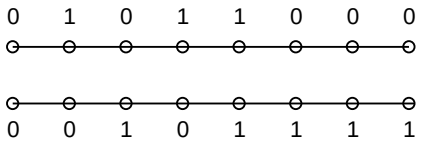


Figure 8.2: Offspring: the result of the mating shown in Fig8.1.

The effect of mating on our example population, with a single crossover site after the first three genes, is given in Table 8.5, and shows that following this evolutionary step, the average fitness has increased from 292.5 to 439.

[2]Two offstrings?

| **s** | $32x$ | $f(x)$ |
|---|---|---|
| 01100 | 12 | 144 |
| 11001 | 25 | 625 |
| 11011 | 27 | 729 |
| 10000 | 16 | 256 |

Table 8.5: After mating: this population forms the next generation, and has average fitness 439.

We now discard the strings from the initial population, replacing them with their "children", and allow the process to be iterated, selection and mating produce further generations, whose fitness we hope continues to increase.

**Mutation** Our initial population was a "good" one in that we can find individuals for which any individual gene is either "on" or "off". We could describe this as having genetic diversity . However the evolutionary process will select against particular genes and allow them to "die out". An example of this is in our first generation population above; it is likely that the first individual in this population, with fitness 144 will not be selected for mating, and hence the first "0" in the gene pool will be eliminated. Since a "1" here increases the value of $x$ by 256, it contributes strongly to the fitness of the string, and we see here the process working, with a "bad" gene being selected against.

As we have described the process so far, no future individual will have the top bit missing. However there is a disadvantage in reducing genetic diversity, as can be seen with the same individual, who also carries the only example of "1" in position 3, which we know to be beneficial in this case. Thus at the point when this gene is eliminated, there is no possibility of evolving to the optimal solution.

To counteract this effect a process of **mutation** is introduced in which, after mating, each bit in the genetic material of the whole of the new population has a very small probability of changing from a "0" to a "1" or vice-versa. This allows new genetic material to be examined; the idea is that if the new gene is "better", it will quickly be adopted by the rest of the population.

## 8.2 Variations

Almost none of the *detail* that has just been described is essential in order to produce a genetic algorithm capable of worthwhile results. For example there are many selection strategies to give the new generation; here are some.

- Discard the least fit 50% of the population and allow each remaining string to breed twice

- Assume string $s_i$ has fitness $f_i$. A string is then selected for breeding with probability $f_i/F$, where $F = \sum f_i$, with the full breeding population selected using $n$ independent trials (*roulette wheel breeding*). Thus the possibility of a string mating with itself, when crossover has no effect, is not excluded.

- As above, but first introduce the fittest string (or the fittest 5% of strings to the breeding pool.
- As in roulette wheel breeding, but the fittest string is selected to mate with the remaining fit strings.

It is clear that many more strategies both for mating and selection can be described, each tailored to overcome difficulties with the naive behaviour. Here are two examples of the type of problem that can occur

**Dominance:** it can happen, usually early in a run, that one string is very much fitter than the others; " a few extraordinary individuals in a population of mediocres."

**Mediocrity:** there can be little difference in fitness between the average and the best individuals, and the algorithm becomes "a random walk among the mediocre".

A solution to both of these problems is to rescale the fitness function; assuming that $f \geq 0$, let $f_1 = af + b$, where $a$ and $b$ are chosen so that $f_1 \geq 0$ and so that the fittest individual is twice as fit as the average.

*8.2. Example.* Use a genetic algorithm to solve the problem:

$$\text{maximise } f(x) = 256\left[\tfrac{1}{4}(x - \tfrac{1}{2})^2\right] \text{ on } [0, 1) \text{ working on a grid of size } 1/32.$$

*Solution* This is relatively straightforward with the correct choice of coding, but a binary coding is not sensible; note that although the maximum occurs when $x = 16/32$ the string corresponding to $x = 15/32$ is also very fit, yet its binary representation differs in every position from that of the fittest string. Evolution from $x = 15/32$ to $x = 16/32$ is thus very unlikely in this coding.

The resolution in this example is to change the coding scheme; this is an example in which the benefits of Gray coding show clearly. However the underlying problem should be noted. We return to it in Section 8.3.

*8.3. Example.* Consider our original problem of maximising $f(x) = 1024x^2$, but now suppose the formula is modified so that $f(0) = 1024$.

*Solution* Such an objective function almost necessarily defeats any advantage the genetic algorithm has. Although the maximum occurs at 0, there is no evidence near 0 that this string is in any way good. Unless the optimal solution is first found by chance, and then is deliberately retained, it will disappear. There is clearly no advantage here in using a genetic algorithm. Of course we can still hope for a "near optimal" solution as is likely to be obtained in this case.

## 8.3 Further Discussion

Clearly the idea of "evolving" a solution to a difficult problem under competitive pressure is theoretically attractive. Having discussed how a genetic algorithm can achieve this in principle, it it time to investigate whether the promise is fulfilled in practice Much of this section is taken from Mitchell (1996), which should be consulted for more details.

Our first example, of $f(x) = 1024x^2$ was easy to do in other ways. It is simple to check that the algorithm given does work, and it is easy to see why that is. However it is much less clear that an advantage accrues to the genetic methodology, as opposed to something less organised. We start by given a number of alternative possible search strategies which could be used on the type of "large discrete search space" problems we have been considering.

**GA:** genetic algorithm as described above.

**SAHC:** steepest ascent hill-climbing:

1. Choose a string at random, and call it the "current hilltop";
2. Systematically flip each bit of the current hilltop and record the resulting fitness;
3. if there is no string with higher fitness, save the current hilltop and go to step 1; otherwise reset the highest value as "current hilltop" and go to step 2.

**NAHC:** next ascent hill climbing:

1. Choose a string at random, and call it the "current hilltop";
2. Systematically flip each bit of the current hilltop until the resulting fitness increase; if there is no increase, return the bit to its original value.
3. If there is no increase in fitness, save the current hilltop and go to step 1.
4. If a string of higher fitness is found, go to step 2, but continue mutating the string from the position last changed.

**RMHC:** random mutation hill climbing:

1. Choose a string at random, and call it the "current hilltop";
2. Choose a location at random and flip that bit of the current hilltop and record the resulting fitness;
3. If there resulting string has higher fitness, reset the value of 'current hilltop".
4. Go to step 2.

In order to compare searches using comparable resources, the process should stop after the same specified number of fitness evaluations.

**The Problem** is chosen to be simple, but to give some advantage to "evolution". The population consists of all binary strings of length 64. The fitness of a string is to be the number of 1's present in "well positioned blocks". We say that a block is "well positioned" if it consists of 8 consecutive 1's, and the first of them occurs in position 1, 9, 17, etc. Blocks of 8 consecutive 1's are not favoured in any other position. To illustrate this function, the fitness of a number of strings is calculated in Table 8.6.

The results, given in Table 8.7 are not an overwhelming endorsement of the genetic algorithm methodology. Testing was for $256,000$ function evaluations unless the method had earlier converged to the maximum possible fitness of 64. Each algorithm was tested 200 times; the mean and median number of runs to convergence is given, together with the standard error ($\sigma/\sqrt{\text{number of runs}}$) in brackets.

| String | Fitness |
|---|---|
| 00010011 00111000 00011111 10001110 00000111 11110000 00101011 00011100 | 0 |
| 11111111 00000000 00000000 00000000 00000000 00000000 00000000 00000000 | 8 |
| 01111111 10000000 00000000 00000000 00000000 00000000 00000000 00000000 | 0 |
| 11111111 11100000 11111111 00000000 00000000 00000000 00000000 00000000 | 16 |
| 11111111 00111111 10000111 11111110 00000000 00000000 11111111 11111111 | 24 |

Table 8.6: A fitness function designed to test the propagation of "well positioned" blocks of 8 copies of 1. Only blocks whose positions reflect the underlying structure of the string are rewarded.

| 200 runs | GA | SAHC | NAHC | RMHC |
|---|---|---|---|---|
| Mean | 61334(2304) | $> 256000(0)$ | $> 256000(0)$ | 6197(186) |
| Median | 54208 | $> 256000$ | $> 256000$ | 5775 |

Table 8.7: Number of function evaluations needed to find the fittest string using different optimising algorithms

This problem was specifically designed to "assist" the methodology of GA's and the testing (Mitchell 1996, page130) was certainly not by an opponent of the method. It is thus clear that they are not an automatic choice, even if the problem appears appropriate.

## 8.4 The Travelling Salesman Problem

We now discuss a classic optimisation problem in which the constraints form an essential part.

> A salesman has to visit $n$ cities and return to his city of origin. Each city has to be visited exactly once, and the distance ( or more generally the cost) of the journey between each pair of cities is known. The problem is to do the **tour** at minimum total cost.

This forms one of a class of problems known as NP-complete problems which are believed to require a computation time $\exp(kn)$, where $k$ is a constant of the problem, for an exact solution. The "standard" method of obtaining an approximate solution uses an optimisation method known as *simulated annealing* which we discuss in Chapter 9. In this chapter we show how it can be formulated as a linear programming problem, and as an example in which genetic optimisation is possible. Although presented in this simple form, there are applications for example to circuit board design equivalent to a tour of 17,000 cities.

### 8.4.1 Linear Programming Formulation

We set up the problem by numbering each of the $n$ cities as $C_1, \ldots, C_n$; a tour is then just a permutation, say $(k_1, k_2, k_3, \ldots, k_n)$ of $(1, \ldots, n)$. We interpret this as the tour which starts at $C_{k_1}$, then goes to $C_{k_2}$ and so on, running through the remaining cities up to and including $C_{k_n}$ before finally completing the tour back at $C_{k_1}$. It is thus clear than we can permute

the permutation cyclically without changing the tour. There is no loss of generality then in assuming that $C_{k_1} = C_1$; it is then clear that there are $(n-1)!$ different tours.[3] This also shows that solving the problem by exhaustive enumeration is infeasible unless there are very few cities.

The problem can be formulated as a linear programming problem, as we now do. Introduce (integer) variables $x_{ij}$ where $x_{ij} = 1$ when $C_j$ is the city immediately following $C_i$ in the tour, and $x_{ij} = 0$ otherwise. Then since exactly one city precedes $C_j$ in the tour we have

$$\sum_{i=1}^{n} x_{ij} = 1, \quad \text{for each } j,$$

while since the tour visits city $C_i$ exactly once, exactly one city must follow $C_i$ and so

$$\sum_{j=1}^{n} x_{ij} = 1, \quad \text{for each } i.$$

Given that the distance (or perhaps the cost) travelling from $C_i$ to $C_j$ is $c_{ij}$, the Travelling Salesman Problem then involves minimising the total cost

$$\sum_{i+1}^{n} \sum_{j+1}^{n} c_{ij} x_{ij}$$

subject to the above constraints, the usual reality constraints that $x_{ij} \geq 0$ and the fact that $x_{ij}$ are in fact either 0 or 1.

However this does not capture the problem exactly; as it stands a pair of disjoint subtours will be acceptable. To cope with this we introduce $n-1$ new variables *integral* $u_2$, $u_3$,...,$u_n$ with $u_i \geq 0$ and $(n-1)^2 - (n-1)$ new constraints as follows:

$$u_i - u_j + n x_{ij} \leq n - 1 \quad \text{for} \quad i, j = 2, 3, \ldots n \quad \text{and} \quad i \neq j. \tag{8.1}$$

**8.4. Proposition.** *Any tour which satisfies all the above constraints does not split into subtours.*

*Proof.* We assume without loss of generality that we have a tour starting at $C_1$; assume there is a subtour, so the tour starting at $C_1$ returns to $C_1$ before visiting all the cities. Then there must be another subtour, which does *not* visit $C_1$ and which includes $r \leq (n-1)$ cities.

We add up the constraints 8.1 for each of the $r$ non-zero $x_{ij}$ on this tour to get $nr \leq (n-1)r$ since the $u_i$ all occur twice and so cancel. This contradictions shows there are no proper subtours. □

**8.5. Proposition.** *Any solution of the Travelling Salesman Problem satisfies all the constraints given above.*

[3] In general we don't identify a tour and the same tour done the opposite way around

*Proof.* It is enough to show we can define variables $u_i$ which satisfy the constraints 8.1. Let $u_i$ be the position on the route at which $C_i$ is visited. Thus if we have a tour of five cities, say

$$C_1 \to C_3 \to C_5 \to C_4 \to C_2 \to C_1$$

then $u_2 = 5$, $u_3 = 2$ and so on.

Consider now the constraint $u_i - u_j + nx_{ij} \leq n - 1$ for fixed $i$ and $j \neq i$. Note that $u_i \leq n$ while $u_j \geq 2$, so if $x_{ij} = 0$ we have $u_i - u_j + nx_{ij} \leq n - 2$ and our inequality holds.

In the remaining case, $x_{ij} = 1$, so $C_j$ is visited immediately after $C_i$ and $u_j = u_i + 1$. Thus

$$u_i - u_j + nx_{ij} = u_i - (u_i + 1) + n = n - 1,$$

and again the constraint holds. □

*8.6. Remark.* We have not discussed linear programming problems with **integral constraints**, the requirement that the variables be integers, anywhere in these notes. Thus we have no way of actually solving the Travelling Salesman Problem in this formulation, although there are good methods; see for example Kolman & Beck (1995, Chapter 4). A theorem of Hoffman and Kruskal may be of interest in this context to show that some of our earlier examples *were* legitimate applications: if the constraint matrix consists entirely of 0's 1's and $-1$'s, then any solution obtained using the standard simplex method will automatically be integral. This means that there was no problem above until the variables $u_i$ were introduced, at which point there was no longer a guarantee that the simplex method would find variables with integer values.

### 8.4.2 Genetic Algorithm Formulation

With this conventional introduction, we consider how the Travelling Salesman Problem can be tackled using a genetic algorithm. This example, and much of the remaining discussion in this section is taken from Michalewicz (1996, Chapter 10).

The two essential ingredients of a genetic algorithm are the availability of a natural representation of the problem and its constraints, which leads to suitable genetic operations, and the availability of a useful fitness function. In this problem, the fitness function is clear; two tours can be evaluated simply by looking at the total cost of each tour; in contrast there is no natural representation. which describes the problem.

Of course a naive binary representation of each tour is available, in which each city is numbered, and a gene is simply an ordered list of city numbers. This representation does not however lend itself to useful genetic operations; a naive mating may not even produce a tour (some city numbers may be invalid), and is unlikely to be a valid tour with no repeat cities.

We are thus led to seek alternative (non-binary) representations which are adapted to this particular problem. Here are some possible representations.

- the ordinal representation, in which the tour $(1, 2, 4, 3, 8, 5, 9, 6, 7)$ is represented as the list $(1, 1, 2, 1, 4, 1, 3, 1, 1)$ in which the code refers to the position of the next city of the list of unused cities.

- the path representation, which was the natural one we used above; and
- the adjacency representations in which the tour $(1, 2, 4, 3, 8, 5, 9, 6, 7)$ is given in the form $(2, 4, 8, 3, 9, 7, 1, 5, 6)$

Even when cross-over is defined, it does not lead to natural "genetic evolution" and so it is necessary to have much more specialised crossover operators, in which the naive cross-over is "repaired". More details of suitable genetic operators can be found in Michalewicz (1996, Chapter 10)

These ideas suggest that local deterministic improvements can also be used during a generation — perhaps the equivalent of the influence of the environment on the individual, so that the assessed fitness of the individual is not the same as its fitness at birth. In practice such **mimetic** algorithms give better results than a purely evolutionary strategy. One way to describe this is that a genetic algorithm can "get to the right hill" but then local hill climbing can improve on the solution.

## 8.5 Example Problems

We describe a number of "real" problems, both mathematical and non mathematical which genetic algorithms proved able to solve successfully.

### 8.5.1 Numerical Optimisation

Our first example is a "straight" numerical optimisation problem. For example, consider the problem

*8.7. Example.* Minimise $f(x) = -\sum_{j=1}^{10} x_j \left( c_j + \ln \frac{x_j}{x_1 + \cdots + x_{10}} \right)$ subject to

$$\begin{aligned} x_1 + 2x_2 + 2x_3 + x_6 + x_{10} &= 2, \\ x_4 + 2x_5 + x_6 + x_7 &= 1, \\ x_3 + x_7 + x_8 + 2x_9 + x_{10} &= 1, \\ x_1, \ldots, x_{10} &\geq 0.000001. \end{aligned}$$

where the constants $c_j$ are given by

$$\begin{array}{lllll} c_1 = -6.089; & c_2 = -17.164; & c_3 = -34.054; & c_4 = -5.914; & c_5 = -24.721; \\ c_6 = -14.986; & c_7 = -24.100; & c_8 = -10.708; & c_9 = -26.662 & c_{10} = -22.179. \end{array}$$

The minimum is not known, but a genetic algorithm did better than the previously known non-genetic minimum, which was obtained by a conventional rather than a genetic algorithm. For details of a number of examples like this, see Michalewicz (1996, Chapter 7); this is their "test case number 2".

### 8.5.2 Credit Card Scoring

Here the aim was to improve on an existing credit card scoring mechanism. Using the existing constraints of their system, but evolving the weights to be placed on various answers

gave a system which was better than the existing one, reducing the loss by 1/2%. In fact it turned out that not all the existing constraints were being obeyed by the existing system; using the actual constraints of the existing system, with a genetic algorithm led to an improvement of 1.5%; and a saving of "a high six figure sum annually." Even this does not reflect the potential saving, since the work was done with historical, and hence "censored" data; it was not possible to look at the subsequent credit history of someone the algorithm scored well but the existing system rejected, since there was no subsequent history.

### 8.5.3 Designing FGGA's

An FPGA — a Field Programmable Gate Array — is a fairly amorphous sort of computer chip. It has much potential, but no character until that has been designed in. Specifically it has no internal logic of its own, until the connections of each particular gate in the chip have been set up, which can be done after manufacture; in other words, in the field. This contrasts eg with a Pentium etc which is designed with a specific function in mind.

A field programmable gate array (FPGA) has a finite number of internal switches, each of which can be either "on" or "off". It has an output, taken as a real number, which depends on a fixed input, and these switch settings. Programming the chip involves choosing suitable settings for these internal switches. A program can be tested by setting the switches and measuring the output from the fixed input; a good program is one in which the measured output is close to a known "ideal" output.

Given the problem, in which we have to choose a binary state of each of a finite number of gates, it is clear how we can encode a program as a binary string. We are given a real number $x$, say, which measures the difference between the output from the program and the desired output; we take the fitness of the corresponding string to be $1/x$; strings with a greater fitness represent better attempts at the required program.

In case the example seems fanciful, I saw it *running* in Sussex University Innovation Centre in October 1997; it was also featured in recent EPSRC publicity. The aim was to do "learn' how to do certain types of speech recognition.

### 8.5.4 The Cocktail Party Effect

This is another example which I found in a recent "New Scientist' (Ngo & Bhadkamkar 1998) in which a genetic algorithm is being used to do noise cancelling so that "clean" signals are presented to a speech recogniser. The aim is to imitate the voice of one of two speakers, when only the mixed signal is available. Once this is done, the voice of one speaker is subtracted from the mixed signal thus giving the voice of the other speaker. Each can then be processed separately.

The assumption that makes this all work is that the two speakers produce uncorrelated signals; the genetic algorithm then evolves a sound wave so that the it is uncorrelated with the difference of original signal and itself.

Cocktail Parties? The ability of the human ear to pick out a single speaker in a room full of speakers is known as the "cocktail party effect!"

You can find more information at `http://web.interval.com/papers/1997-062`

### 8.5.5 Mondrians

This was an image processing exercise. the aim was to locate grey rectangles on a (slightly different) grey background, in the presence of noise in a square image of size $512 \times 512$. A potential solution could be described by given a small number of parameters representing the grey-scale value (between 0 and 255) of each rectangle, the location of two opposite corners, and its precedence (from front to back say, with higher precedence rectangles appearing nearer the front, and so less likely to be obscured). The "background" is then coded as a full-sized rectangle with the lowest precedence. The fitness function was then derived from how well such a model of the image fitted the (synthetically created) data.

In practice a genetic algorithm, with the "obvious" coding except for the use of Gray, rather than binary codes, proved better than many more elaborate methods based on following edges and guessing intersections.

### 8.5.6 Prisoners Dilemma

This is a classic example of a non-zero sum game and illustrates the conflicts which can arise between the principles of competition and co-operation. The payoff matrix from Rowman's viewpoint is shown in Table 8.8.

| | Columnman remains silent | Columnman defects |
|---|---|---|
| Rowman remains silent | 3,3 | 0,5 |
| Rowman defects | 5,0 | 1,1 |

Table 8.8: Payoff matrix for the Prisoner's dilemma; the return to Rowman is given first, then the return to Columnman

The story behind this which gives the problem its name is that Rowman and Columnman are both prisoners, who were involved in the same enterprise before being put in jail, but are now unable to communicate with each other. Each has been invited to give evidence against the other; the payoff matrix gives the number of years by which their sentence will be reduced given varying outcomes. If one of the prisoners defects and gives evidence against the other, he is rewarded; but only if the other does not make the same choice. Thus the total return to both prisoners is greatest (at 6 years reduction in total sentence) if each remains silent, while the strategy that minimises possible loss whatever the other player does, results in the least total return to both prisoners of 2 years remission.

The problem becomes one of finding the best strategy in a series of games against the same opponent. Theoretical analysis is difficult and in work by Axelrod (Mitchell 1996, Page 30) a number of programs were pitted against each other in tournaments, using many different algorithms. Perhaps surprisingly, a very simple algorithm proved very effective, namely:

> first co-operate (ie remain silent) and then "tit for tat".

The effect is to co-operate as soon as the other program does so. It turns out that genetic algorithms were able to discover this strategy; and indeed even improve on it slightly, by initially defecting, and then "apologising"!

## 8.6 Multiple Objective Functions

One final point should be made, which is valid for any optimisation method. What happens when, as is usual in real life, there is more than one objective function to be optimised, or alternatively, if the constraints which are required cannot be met? Consider first the problem

$$\text{maximise } f_1(\mathbf{x}) \text{ and } f_2(\mathbf{x}) \text{ subject to } g_i(\mathbf{x}) = 0 \text{ for } i \leq i \leq n.$$

Of course in simple cases, maximising $f_1$ may also maximise $f_2$, but the usual situation would involve a compromise. It is necessary to weigh together the two functions and decide how they should be balanced. One solution, indicating a more general approach, simply builds a single objective function from the given ones. Thus the problem might be rewritten as

$$\text{maximise } f_1^2(\mathbf{x}) + f_2^2(\mathbf{x}) \text{ subject to } g_i(\mathbf{x}) = 0 \text{ for } i \leq i \leq n;$$

in exchange for a more complicated objective function, now necessarily quadratic, there is only one thing to optimise. This quadratic objective function is often thought of as a "distance" from the true solution. It is still necessary to see that the problem scales correctly, so that, for example a very large departure from optimising $f_1$ is appropriately reflected in the behaviour of the distance or "penalty" function, and this may introduce further non-linearities. Note also that even if the original functions and constraints were all linear, we move away from this domain by introducing the penalty function.

As alternative way to employ this idea is when the constraints are inconsistent. Consider the problem

$$\text{maximise } f(\mathbf{x}) \text{ subject to } g_i(\mathbf{x}) = 0 \text{ for } i \leq i \leq n,$$

and suppose it turns out that the constraints over determine the problem and there is no solution. We can allow the constraints to be relaxed by minimising a single objective function of the form

$$F(\mathbf{x}) = -f(\mathbf{x}) + \lambda_i(g_1(\mathbf{x})^2 + \lambda_2 g_2(\mathbf{x})^2 + \cdots + \lambda_n g_n(\mathbf{x})^2.$$

Again we necessarily pass to a quadratic function, and again we have to ensure that the scaling parameters $\lambda_1, \ldots, \lambda_n$ are sensibly chosen. This function does not even reproduce the original result if the constraints can always be satisfied, although such a failure probably indicates an inappropriate choice of the $\lambda$'s. A familiar example of this approach is the least square approximation to a straight line through many points; one we met earlier is the the "big $M$" method of Section 4.3.1.

# Chapter 9

# Simulated Annealing

## 9.1 Introduction

Optimisation problems occur frequently in many disciplines, often as a subsidiary task before the interesting problem. As an example assume we wish to investigate the bending of a metal sheet in order to touch (spot weld at?) a collection of non coplanar point. The first step is to fit the best plane to the given points, since this is the bit that can be done with no bending, and then look at the bending necessary from this equilibrium situation in order to achieve the required result.

Standard calculus methods will work here. We specify the plane with three parameters, and then use the normal equations to calculate the best fit. Recall that we derived normal equations by looking for critical points of the fitting error, regarded as a function of the parameters defining the plane. The problem is almost always well behaved, and we quickly come up with a solution before getting on with the real work.

More generally, iterative methods have traditionally been calculus-based. The **conjugate gradient** method computes not just the value of the function at a given point, but also the value of the derivative there. In several dimensions, this then gives information about which way to move in order to improve (decrease if we are minimising) the value of the function. One "obvious", but not very good way is the method of **steepest descent** in which we minimise down the line of maximum gradient, move to that point and repeat. A good overview of many such methods is in Press et al. (1992, Chapter 10); in particular it describes the problems with the steepest descent methods(Press et al. 1992, Page 421) in a long narrow valley.

All such traditional optimisation methods have a problem in the presence of functions which have many local minima in the area of concern. One such "difficult" example, even in one dimension, is shown in Fig. 9.1. It isn't even artificial; the function is $f(x) = (2 + \sin(\pi x)) \cos(20\pi x)$, and naturally crops up when studying amplitude modulation; $f$ is "really" the rapidly varying function $\cos(20\pi x)$, but the amplitude has been modulated by the slowly varying function $2 + \sin(\pi x)$. If this looks the sort of function you can easily deal with, how about Fig 9.2, where at coarse scale there appears to be a simple minimum, while at the scale drawn, it is clear there is a small fluctuation which confuses the overall picture. The function is $(1 + \cos(30\pi x)/15) \cos(\pi x)$; almost the same as $\cos(\pi x)$ but with the additional "waviness" $\cos(30\pi x)/15$ which causes the trouble.

This chapter complements the work on genetic algorithms in Chapter 8 and provides a

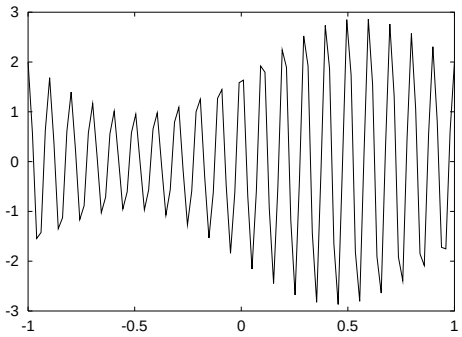


Figure 9.1: Locating a global maximum can be hard.

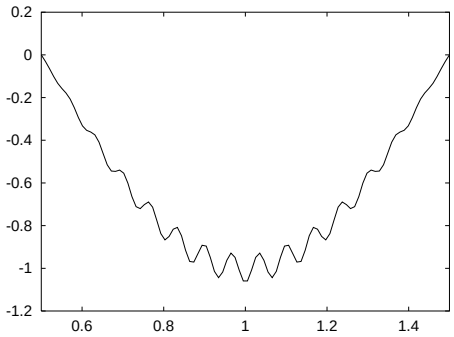


Figure 9.2: An apparently smooth minimum may be like this?

second example of what could be considered a *modern* optimisation method. Such methods take advantage of the computing power now easily available on the desk, and go beyond traditional techniques which require significant mathematical analysis to understand the situation before starting a relatively circumscribed numerical process.

### 9.1.1 Overview

The process of **annealing** is one in which a solid, usually metal, is first heated, and then allowed to cool slowly. As the solid cools, a change of state takes place in which individual atoms arrange themselves into a regular array corresponding to a minimum energy arrangement. Such an arrangement cannot easily propagate throughout the solid if the cooling occurs quickly, and boundaries between different "domains of regularity" occur. Such boundaries introduce potential "fault-lines" along which a fracture is most likely to occur when the material is stressed. To avoid such potential failures, metal is often cooled slowly, in a process known as *annealing* to permit re-arrangements at these boundaries so the *same* local minimum energy arrangement occurs throughout the material.

This process is imitated in numerical optimisation. The idea originated with Metropolis, Rosenbluth, Rosenbluth, Teller & Teller (1953) when trying to simulate such thermodynamic systems. Given a potential state change from one with energy $E_1$ to one with energy $E_2$, they chose to accept it with a probability

$$\text{Prob( Accept)} = \min(1, \exp\{-(E_2 - E_1)/kT\})$$

where $T$ is the "temperature" and $k$ is a constant — in this application it is Boltzmann's constant. In words this always accepts a change if it moves to a state of lower energy, but *sometimes* accepts the change even though the system moves to a state with a higher energy. Note that for small $T$ there is a very small probability of accepting an unfavourable move, while for large $T$, the probability of acceptance can be quite high.

With this in mind we now describe the requirements in order to apply the same ideas to a more general minimisation problem. We need

- a coding of the possible system states;
- an objective function which we are trying to minimise;

- a mechanism for proposing random changes to the state of the system; and
- A control parameter $T$, analogous to the temperature above, which governs the probability of acceptance of the proposed change, together with an **annealing schedule** specifying how the temperature is to be lowered.

## 9.2 An Example - the Travelling Salesman Problem

We illustrate these ideas by applying them to the travelling salesman problem described in Section 8.4. This example is given in Press et al. (1992, Section 10.9).

### Coding

There is no need for a carefully tailored coding scheme because we aren't going to combine bits of it "at random" as we did with the genetic algorithm. Instead, we can simply use the obvious path representation described in Section 8.4.2 on page 96. Thus each potential tour can be described by a re-arrangement $\mathbf{i}$ of the city labels.

### Re-arrangements

We try to generate "intelligent" changes, and indeed this is at the heart of the method. Here are two possibilities:

- a section of the tour is cut out and replaced by one running the other way round; or
- the section is replaced by a random re-arrangement of that section.

### Objective Function

In its simplest form, this is straightforward. In a "real" problem we know the distances between each pair of cities, so we simply minimise the objective function

$$f(\mathbf{i}) = \sum_{\mathbf{i}} \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2},$$

where $(x_i, y_i)$ is the position of the $i^{\text{th}}$ city. More generally the "distances" can be travelling costs etc. It is possible to manipulate the answer eg by adding an additional cost to go between different "regions"; this for example could bias keeping the tour in Scotland until all Scottish cities have been done, before moving to England etc.

### Annealing Schedule

Here I am very dependent on the experience of others. But this is an outline of how you might start. The first need is to get the scale of the problem correct - try some typical solutions to find out how the objective function varies, and use this to choose the parameter $T$ so that initially it will allow such changes with high probability. Then decrease $T$ by (say) 10% each time as part of the cooling schedule, and run the algorithm until either

- a total of $100N$ reconfigurations have occurred; or

- a total of $10N$ successful reconfigurations have occurred.

Next, decrease the temperature by 10% and repeat the calculations. Continue in this way until there is so little change that the system seems to have converged.

## 9.3 Minimising a Function

Our running example is that of minimising an objective function $f(\mathbf{x})$; we conclude by seeing how the necessary ingredients are obtained. Of course the objective function $f$ is clear, while the control parameter $T$ and the cooling schedule have to be set by experiment as before. The state of the system is given by the variable $\mathbf{x}$, so to run the algorithm we simple have to describe how to move from a potential solution $\mathbf{x}$ to a new one $\mathbf{x} + \delta\mathbf{x}$. One such method is as follows; calculate the gradient at $\mathbf{x}$, and move downhill a distance which depends on the value of the gradient, so small moves are made if the function appears nearly flat. In fact there are many things wrong with our basic "change of state" algorithm; the problems are described by (Press et al. 1992, Page 451). You should go there for more information.

## 9.4 Comparison

Finally we compare this method of solution with that using the genetic algorithm. Unlike many other "intelligent" algorithms, simulated annealing is not *greedy* in the sense that it always accepts the best local move. It also seems capable of making decisions in a logical order, first getting the gross features of the problem (ie at high temperature) and only subsequently refining them. In contrast, the genetic algorithm seems much more like random search; in fact one is totally dependent on the coding being closely connected with the essence of the problem. However a genetic algorithm is almost trivial to program and so is capable of producing results quite quickly.

Either can work. My own guess is that if you know enough about the problem to try simulated annealing, it is likely to provide a better solution, but I find the results of applying a genetic algorithm to the travelling salesman problem are impressive.

## 9.5 Questions 7 (Hints and solutions start on page 127.)

*9.1. Q.* Describe the various components of a simple genetic algorithm and explain how such an algorthim can be used to maximise a function. Illustrate your answer by considering the problem:

$$\text{maximise } \{n^2 \mid n \quad \text{is an integer with } 0 \leq n \leq 31\}.$$

*9.2. Q.* A field programmable gate array (FPGA) has a finite number of internal switches, each of which can be either "on" or "off". It has an output, taken as a real number, which depends on a fixed input, and these switch settings. Programming the chip involves choosing suitable settings for these internal switches. A program can be tested by setting the switches and measuring the output from the fixed input; a good program is one in which the measured output is close to a known "ideal" output.

Explain how a genetic algorithm could be used to evolve a good program for an FPGA, indicating briefly the processes involved.

*9.3. Q.* Write a comparison of linear and non-linear optimisation methods. You should give a brief indication of typical methodologies in each case (but for just *one* non-linear method), and describe the type of problem for which each method is suitable. Comment on performance issues in general, and indicate the performance you would expect from each method, if both methods could be applied to a particular problem.

*9.4. Q.* Describe briefly methods you might use for a non-linear optimisation problem, giving an indication of typical methodologies both for problems which are "nearly" linear and more general ones. Comment on performance issues in general, and indicate the performance you would expect from each method, if both methods could be applied to a particular problem.

*9.5. Q.* a) Describe briefly the Travelling Salesman Problem. Illustrate your description by explicitly finding the shortest tour given the distances (or costs) between cities shown in Table 9.1.

| To / From | A | B | C | D |
|---|---|---|---|---|
| A | - | 4 | 6 | 12 |
| B | 3 | - | 6 | 8 |
| C | 7 | 10 | - | 10 |
| D | 11 | 7 | 9 | - |

Table 9.1: Distances (or costs) between cities.

b) Describe briefly how a solution to the Travelling Salesman Problem might be obtained using a genetic algorithm, indicating how features of the problem map to the elements needed to use a genetic algorithm.

c) Describe briefly how a solution to the Travelling Salesman Problem might be obtained using simulated annealing, again indicating how you would obtain the necessary features for *this* problem.

# Appendix A

# Solutions to Exercises

## Solutions for Questions 1 (page 10).

***Solution*** 1.1:

a) Smaller rolls can be created using a number of methods. In method $M_1$, the roll is cut into two equal 9 ft. rolls with no waste. In method $M_2$ the roll is cut into a 9 ft. and a 7 ft. roll with 2 ft. of waste. In method $M_3$ the roll is cut into a 9 ft. and a 5 ft. roll with 4 ft. of waste. In method $M_4$ the roll is cut into two 7 ft. rolls with 4 ft. of waste. In method $M_5$ the roll is cut into a 7 ft. and two 5 ft. rolls with 1 ft. of waste. Finally in method $M_6$ the roll is cut into three 5 ft. rolls with 3 ft. of waste.

b) Let $x_i$ be the number of large rolls that are cut using method $M_i$. Then there are $2x_1+x_2+x_3$ rolls available that are 9 ft long, $x_2 + 2x_4 + x_5$ rolls available that are 7 ft long and $x_3 + 2x_5 + 3x_6$ rolls available that are 5 ft long. Thus the linear programming problem becomes:

$$\text{minimise } x_1 + x_2 + x_3 + x_4 + x_5 + x_6, \text{ subject to } x_i \geq 0 \quad (1 \leq i \leq 6) \text{ and}$$

$$\begin{aligned} 2x_1 + x_2 + x_3 &\geq 10, \\ x_2 + 2x_4 + x_5 &\geq 20, \\ x_3 + 2x_5 + 3x_6 &\geq 50. \end{aligned}$$

There are at least two problems that could occur in practice. One is the possibility that $x_i$ may not be integers; clearly it would be necessary to use the next largest integer number of rolls. The second is the difficulty in making the cut using method $M_1$, which involves no waste at all; in practice the resulting rolls may be too short. It may be that all three constraints above are tight, so we create exactly the right the number of smaller rolls of each size; if not some account should be taken of that waste.

c) There are a total of $x_1 + 2(x_2 + x_3 + x_4) + 3(x_5 + x_6)$ cuts to be made, while the amount of waste in feet is $2x_2 + 4(x_3 + x_4) + x_5 + 3x_6$. Thus the total net cost $N$ in pounds is

$$\begin{aligned} N = (x_1 + x_2 + x_3 + x_4 + x_5 + x_6)P \\ + (x_1 + 2x_2 + 2x_3 + 2x_4 + 3x_5 + 3x_6)C \\ - (2x_2 + 4x_3 + 4x_4 + x_5 + 3x_6)k. \end{aligned}$$

and this gives the objective function to minimise subject to the same constraints as above.

***Solution*** 1.2:

a) Let the factory produce $b$ copies of the "Bashful" sculpture, $d$ copies of the "Dozy" sculpture and $h$ copies of the "Happy" sculpture each week. The "reality" condition insists that $b \geq 0$, $d \geq 0$ and $h \geq 0$ — and to be really formal, that each of $b$, $d$ and $h$ is integral. The total profit $P$ in £'s is given by

$$P = 2b + 4d + 3h.$$

We have constraints based on the availability of the machines. From the given table, we will use $2b + d$ hours on machine $A$, $b + 3h$ hours on machine $B$ and $2b + 3d + 2h$ hours on machine $C$. Thus our availability constraints are:-

$$\begin{aligned} 2b + d &\leq 43, \\ b + 3h &\leq 37, \\ 2b + 3d + 2h &\leq 42. \end{aligned}$$

All the constraints are linear; hence the formulation as a linear programming problem is simply to maximise $P$ subject to these three constraints and the reality constraint.

***Solution*** 1.3: Let $b$, $d$ and $f$ be the number of kilograms of binder, disintegrant and filler in each 100 kilograms of the formulation. Then since there will be 14 kilograms of active ingredient in each 100 kilograms of the formulation, $b + d + f = 86$. The binder - filler constraint is that $10b \geq f$, while the constraint on the disintegrant gives $4d \leq b + 14$. These, together with the reality requirement, that $b \geq 0$, $d \geq 0$ and $f \geq 0$ are all the constraints, and the problem is to minimise the total cost $C = 50b + 15d + 2f$ subject to these constraints.

***Solution*** 1.4: Let $a$, $b$ and $c$ be the number of cars of each type that are to be made. The reality constraint, that $a \geq 0$, $b \geq 0$ and $c \geq 0$ is clearly essential. The labour availabilty in the two factories gives:

$$\begin{aligned} 8a + 8b + 9c &\leq 10120, \\ 8a + 9b + 11c &\leq 11000 \end{aligned}$$

An aditional constraint might be the need for $a$, $b$ and $c$ to be integers, although since this is a monthly figure, it would be natural to hold uncompleted cars until the following month. The total profit made $P$, in pounds, is then $P = 1100a + 1200b + 1450c$.

The mathematical model is unrealistic in many respects. Some factors are:

- it is very unlikely that the assumption of constant profit per vehicle is true; there are probably some fixed costs involved and also capacity problems.
- there is no reflection of market demand; it *is* plausible that no cars of type $B$ are made, a result which would be unacceptable in practice; and
- there is an unlikely simplicity in the product range; I would expect there to be many more options with varying nett profits in a real situation.

***Solution*** 1.5: Let $r_1$, $r_2$ and $r_3$ be the number of kilos of cereal, dried fruit and nuts respectively which are mixed to make the "Rich" blend, and define $h_1$, $h_2$ and $h_3$ and $c_1$, $c_2$ and $c_3$ to be the corresponding weights for the "Healthy" and "Crunchy" mixes.

The total costs of the cereals is

$$C = 1.5(c_2 + h_2 + r_2) + 1.0(c_3 + h_3 + r_3) + 0.8(r_1 + h_1 + c_1)$$

while the total sales income is

$$S = 2.0(r_1 + r_2 + r_3) + 1.6(c_1 + c_2 + c_3) + 1.2(h_1 + h_2 + h_2)$$

and the difference $S - C$ between these two figures gives the profit which is to be maximised.

The constraint that the "Crunchy" blend must contain at least 60% nuts becomes

$$c_3 \geq 0.6(c_1 + c_2 + c_3).$$

The other constraints are given in the same way: for the "Healthy" mix,

$$h_1 \geq 0.6(h_1 + h_2 + h_3) \quad \text{and} \quad h_3 \leq 0.2(h_1 + h_2 + h_3)$$

and for the "Rich" blend,

$$r_1 \leq 0.2(c_1 + r_2 + r_3) \quad \text{and} \quad r_2 \geq 0.6(c_1 + r_2 + r_3).$$

We have three supply constraints

$$\begin{aligned} c_1 + h_1 + r_1 &\leq 100, \\ c_2 + h_2 + r_2 &\leq 80, \\ c_3 + h_3 + r_3 &\leq 60. \end{aligned}$$

In addition of course we have the reality constraints that $h_i \geq 0$, $c_i \geq 0$ and $r_i \geq 0$.

## Solutions for Questions 2 (page 27).

***Solution*** 2.1: We first introduce slack variables and convert to tableau form.

```
> with(linalg):    A:=matrix(3,2,[1,-1,2,1,-5,-2]):
> B:=concat(A,diag(1,1,1),vector([2,7,0]));
```

$$B := \begin{bmatrix} 1 & -1 & 1 & 0 & 0 & 2 \\ 2 & 1 & 0 & 1 & 0 & 7 \\ -5 & -2 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Since this is a maximising problem, there is a choice of two proper signs for improvement, in columns 1 and 2. Choosing column 2 means there is no choice of ratios, since one of the two relevant entries is negative and so ignored.

```
> C:=pivot(B,2,2);
```

$$C := \begin{bmatrix} 3 & 0 & 1 & 1 & 0 & 9 \\ 2 & 1 & 0 & 1 & 0 & 7 \\ -1 & 0 & 0 & 2 & 1 & 14 \end{bmatrix}$$

After pivoting there is a proper sign for improvement in column 1. Looking at the appropriate ratios shows we should swap $\mathbf{a}_3$ out from the basis.

```
> E:=mulrow(C,1,1/3):F:=pivot(E,1,1);
```

$$F := \begin{bmatrix} 1 & 0 & \frac{1}{3} & \frac{1}{3} & 0 & 3 \\ 0 & 1 & \frac{-2}{3} & \frac{1}{3} & 0 & 1 \\ 0 & 0 & \frac{1}{3} & \frac{7}{3} & 1 & 17 \end{bmatrix}$$

Making the other choice initially leads to exactly the same choices, done in the opposite order.

```
> C:=pivot(B,1,1);E:=mulrow(C,2,1/3):F:=pivot(E,2,2);
```

$$C := \begin{bmatrix} 1 & -1 & 1 & 0 & 0 & 2 \\ 0 & 3 & -2 & 1 & 0 & 3 \\ 0 & -7 & 5 & 0 & 1 & 10 \end{bmatrix}$$

$$F := \begin{bmatrix} 1 & 0 & \frac{1}{3} & \frac{1}{3} & 0 & 3 \\ 0 & 1 & \frac{-2}{3} & \frac{1}{3} & 0 & 1 \\ 0 & 0 & \frac{1}{3} & \frac{7}{3} & 1 & 17 \end{bmatrix}$$

The maximum value is 17 which occurs when $x_1 = 3$ and $x_2 = 1$.

***Solution*** 2.2: We first introduce slack variables and convert to tableau form.

```
> with(linalg):    A:=matrix(3,2,[-2,1,1,3,3,-4]):
> B:=concat(A,diag(1,1,1),vector([5,22,0]));
```

$$B := \begin{bmatrix} -2 & 1 & 1 & 0 & 0 & 5 \\ 1 & 3 & 0 & 1 & 0 & 22 \\ 3 & -4 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Since this is a maximising problem, there is only one proper sign for improvement, in column 2. Looking at the appropriate ratios shows we should swap $\mathbf{a}_3$ out from the basis. Pivoting, we have:-

```
> C:=pivot(B,1,2);
```

$$C := \begin{bmatrix} -2 & 1 & 1 & 0 & 0 & 5 \\ 7 & 0 & -3 & 1 & 0 & 7 \\ -5 & 0 & 4 & 0 & 1 & 20 \end{bmatrix}$$

Again there is only one proper sign for improvement. This time there is no choice of ratios, since one of the two relevant entries is negative and so ignored.

```
> E:=mulrow(C,2,1/7);
```

$$E := \begin{bmatrix} -2 & 1 & 1 & 0 & 0 & 5 \\ 1 & 0 & \frac{-3}{7} & \frac{1}{7} & 0 & 1 \\ -5 & 0 & 4 & 0 & 1 & 20 \end{bmatrix}$$

```
> F:=pivot(E,2,1);
```

$$F := \begin{bmatrix} 0 & 1 & \frac{1}{7} & \frac{2}{7} & 0 & 7 \\ 1 & 0 & \frac{-3}{7} & \frac{1}{7} & 0 & 1 \\ 0 & 0 & \frac{13}{7} & \frac{5}{7} & 1 & 25 \end{bmatrix}$$

There are now no proper signs for improvement, showing the algorithm has converged. the maximum value is 25 and this is attained when $x_1 = 1$ and $x_2 = 7$.

***Solution*** 2.3: We first introduce slack variables and convert to tableau form.

```
> with(linalg):    A:=matrix(3,2,[1,-2,2,1,-3,1]):
> B:=concat(A,diag(1,1,1),vector([5,15,0]));
```

$$B := \begin{bmatrix} 1 & -2 & 1 & 0 & 0 & 5 \\ 2 & 1 & 0 & 1 & 0 & 15 \\ -3 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Since this is a maximising problem, there is only one proper sign for improvement, in column 1. Looking at the appropriate ratios shows we should swap $\mathbf{a}_3$ out from the basis.

```
> C:=pivot(B,1,1);
```

$$C := \begin{bmatrix} 1 & -2 & 1 & 0 & 0 & 5 \\ 0 & 5 & -2 & 1 & 0 & 5 \\ 0 & -5 & 3 & 0 & 1 & 15 \end{bmatrix}$$

Again there is only one proper sign for improvement. This time there is no choice of ratios, since one of the two relevant entries is negative and so ignored.

```
> E:=mulrow(C,2,1/5);
```

$$E := \begin{bmatrix} 1 & -2 & 1 & 0 & 0 & 5 \\ 0 & 1 & \frac{-2}{5} & \frac{1}{5} & 0 & 1 \\ 0 & -5 & 3 & 0 & 1 & 15 \end{bmatrix}$$

```
> F:=pivot(E,2,2);
```

$$F := \begin{bmatrix} 1 & 0 & \frac{1}{5} & \frac{2}{5} & 0 & 7 \\ 0 & 1 & \frac{-2}{5} & \frac{1}{5} & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 20 \end{bmatrix}$$

There are now no proper signs for improvement, showing the algorithm has converged. the maximum value is 20 and this is attained when $x_1 = 7$ and $x_2 = 1$.

***Solution*** 2.4: We first introduce slack variables and convert to tableau form.

```
> A:=matrix(3,3,[3,-2,3,-2,1,6,1,3,-1]):
> B:=concat(A,diag(1,1,1),vector([3,5,0]));
```

$$B := \begin{bmatrix} 3 & -2 & 3 & 1 & 0 & 0 & 3 \\ -2 & 1 & 6 & 0 & 1 & 0 & 5 \\ 1 & 3 & -1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

This is a minimising problem; we thus swap in either $\mathbf{a}_1$ or $\mathbf{a}_2$ since the corresponding entries in the last row have the proper sign for improvement. With either choice, there is only one positive entry in the column about which to pivot. Technically it is easier to pivot about the 1 in $\mathbf{a}_2$. This gives

```
> C:=pivot(B,2,2);
```

$$C := \begin{bmatrix} -1 & 0 & 15 & 1 & 2 & 0 & 13 \\ -2 & 1 & 6 & 0 & 1 & 0 & 5 \\ 7 & 0 & -19 & 0 & -3 & 1 & -15 \end{bmatrix}$$

Since now $\mathbf{a}_1$ has the proper sign for improvement, but each entry in that column (apart from the objective row) is negative, we deduce that there is no minimum.

Doing the first pivot about $\mathbf{a}_1$ gives essentially the same result; in this case the new $\mathbf{a}_2$ has the proper sign for improvement but each entry in that column (apart from the objective row) is negative.

```
> C:=pivot(B,1,1);
```

$$C := \begin{bmatrix} 3 & -2 & 3 & 1 & 0 & 0 & 3 \\ 0 & \frac{-1}{3} & 8 & \frac{2}{3} & 1 & 0 & 7 \\ 0 & \frac{11}{3} & -2 & \frac{-1}{3} & 0 & 1 & -1 \end{bmatrix}$$

***Solution*** 2.5: Let $B = \{\mathbf{v}_1, \mathbf{v}_2 \ldots, \mathbf{v}_n\}$ be the given basis and let $B' = B \setminus \{\mathbf{v}_1\} \bigcup \{\mathbf{u}\}$. Then a necessary and sufficient condition that $B'$ is a basis is that $\lambda_1 \neq 0$.

To see this, suppose first that $\lambda_1$ is nonzero. In order to show that $B'$ is a basis, it is enough to show it is linearly independent, since it has the same number of elements as the basis $B$. So assume that

$$\mu\mathbf{u} + \sum_{i=2}^{n} \mu_i \mathbf{v}_i = \mathbf{0};$$

we show that each coefficient vanishes. From the given expression for $\mathbf{u}$ in terms of $B$, we have

$$\mu\lambda_1\mathbf{v_1} + \sum_{i=2}^{n} (\mu\lambda_i + \mu_i)\mathbf{v}_i = \mathbf{0},$$

and since this is a linear relation between members of the basis $B$, each coefficient must vanish. In particular, since $\lambda_1 \neq 0$, we have $\mu = 0$, and so also $\mu_i = 0$ for all $i$. Thus the set $B'$ is linearly independent.

Conversely, suppose that $B'$ is a basis. Then in particlar it spans the space, so we can write

$$\mathbf{v} = \mu\mathbf{u} + \sum_{i=2}^{n} \mu_i \mathbf{v}_i,$$

and $\mu$ is non-zero, since otherwise we have

$$\mathbf{v} = \sum_{i=1}^{n} \mu_i \mathbf{v}_i,$$

and this is a non-trivial linear combination of (linearly independent) elements from the basis $B$, which cannot occur.

Since $\mu$ is non-zero, we can write

$$\mathbf{u} = \frac{1}{\mu}\mathbf{v} - \sum_{i=2}^{n} \frac{\mu_i}{\mu}\mathbf{v}_i,$$

and the coefficient $\frac{1}{\mu}$ is non-zero as required.

## Solutions for Questions 3 (page 53).

***Solution*** 4.1:

a) In this part of the algorithm, it is necessary to pivot about the element $y_{q+1,s}$ in order to maintain a feasible tableau, and increase the value of $u_{q+1}$ in an attempt to make it become positive. Pivoting, we have

| Basis | $\mathbf{a}_1$ | ... | $\mathbf{a}_s$ | ... | $\mathbf{e}$ | $\mathbf{b}$ |
|---|---|---|---|---|---|---|
| $\mathbf{a}_{h_1}$ | $y_{1,1}$ | ... | $y_{1,s}$ | ... | 0 | $u_1 \quad (>0)$ |
| $\vdots$ | | | $\vdots$ | | $\vdots$ | $\vdots$ |
| $\mathbf{a}_{h_q}$ | | | $y_{q,s}$ | | 0 | $u_q \quad (>0)$ |
| $\mathbf{a}_{h_{q+1}}$ | $y_{q+1,1}$ | ... | $y_{q+1,s}$ | ... | 0 | $u_{q+1} \quad (<0)$ |
| $\mathbf{e}$ | $v_1$ | ... | $v_s$ | ... | 1 | $z$ |
| $\mathbf{a}_{h_1}$ | $*$ | ... | 0 | ... | 0 | $u_1 - y_{1,s}(u_{q+1}/y_{q+1,s})$ |
| $\vdots$ | | | $\vdots$ | | $\vdots$ | $\vdots$ |
| $\mathbf{a}_{h_q}$ | | | 0 | | 0 | $u_q - y_{q,s}(u_{q+1}/y_{q+1,s})$ |
| $\mathbf{a}_s$ | $*$ | ... | 1 | ... | 0 | $u_{q+1}/y_{q+1,s}$ |
| $\mathbf{e}$ | $*$ | ... | 0 | ... | 1 | $z - v_s(u_{q+1}/y_{q+1,s})$ |

We have written $*$ for entries which are irrelevant. We now show that each entry in the column headed $\mathbf{b}$ is $\geq 0$, apart from the last entry which is the value of the objective function. This will show that the new tableau corresponds to a feasible solution. Note first that since both $y_{q+1,s}$ and $u_{q+1}$ are negative, the new value in this column, (ie the value of $x_s$), is $u_{q+1}/y_{q+1,s} > 0$. Each of the other basic variables is modified by the addition of $-y_{r,s}(u_{q+1}/y_{q+1,s})$, and since we are given that $-y_{r,s} > 0$, it follows that each good row remains good.

b) We first introduce an artificial variable $x_5$ and slack variable $x_6$. Since we want an initial tableau with a basic feasible solution, we rewrite the second inequality so it appears as a bad row.

```
> A:=matrix(3,4,[1,1,0,3,-1,-1,1,-2,-6,-7,-1,-15]):
> B:=concat(A,diag(1,1,1),vector([6,-5,0]));
```

$$B := \begin{bmatrix} 1 & 1 & 0 & 3 & 1 & 0 & 0 & 6 \\ -1 & -1 & 1 & -2 & 0 & 1 & 0 & -5 \\ -6 & -7 & -1 & -15 & 0 & 0 & 1 & 0 \end{bmatrix}$$

However, out first aim is to remove the artificial variable from the basis. We can pivot about any non-zero entry in the first row; we choose to pivot about the first such.

```
> C:=pivot(B,1,1);
```

$$C := \begin{bmatrix} 1 & 1 & 0 & 3 & 1 & 0 & 0 & 6 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & -1 & -1 & 3 & 6 & 0 & 1 & 36 \end{bmatrix}$$

From now on, since $x_5$ is a non-basic variable, we ignore column 5. Since we are minimising, there is a proper sign for improvement in column 4. Examining the relevant ratios, shows we must pivot about the entry in row 2, giving:

```
> E:=pivot(C,2,4);
```

$$E := \begin{bmatrix} 1 & 1 & -3 & 0 & -2 & -3 & 0 & 3 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & -1 & -4 & 0 & 3 & -3 & 1 & 33 \end{bmatrix}$$

The resulting tableau has no proper sign for improvement, so the minimum value is 33, which occurs when $x_1 = 3$ and $x_4 = 1$. Of course we can reduce the objective function using column 5, but this will result in the artificial variable being non-zero; thus an infeasible solution.

***Solution*** 4.2: If the problem is unbounded, eventually a column will occur in which the row corresponding to the objective function will have the proper sign for improvement, but everything above that column will be an inappropriate pivot, in that each entry is negative or zero. Even if there are appropriate pivots in other columns, this, by itself, shows that the problem is unbounded in that suitable multiples of the solution corresponding to that column can be added to the existing solution to make it as large (in the required sense) as chosen.

To solve the given problem we first introduce slack variables. In doing this we see that the second constraint generates a bad row; we move it down so it comes after all the good rows.

```
> with(linalg):    A:=matrix(4,3,[1,1,-2,1,3,-8,4,-1,1,6,-2,3]):
> B:=concat(A,diag(1,1,1,1),vector([2,8,-6,0]));
```

$$B := \begin{bmatrix} 1 & 1 & -2 & 1 & 0 & 0 & 0 & 2 \\ 1 & 3 & -8 & 0 & 1 & 0 & 0 & 8 \\ 4 & -1 & 1 & 0 & 0 & 1 & 0 & -6 \\ 6 & -2 & 3 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Treating the bad row as the objective function, there is only one proper sign for improvement, in column 2; the corresponding ratios are 2 and 8/3 so we swap $\mathbf{a}_4$ out of the basis.

```
> C:=pivot(B,1,2);
```

$$C := \begin{bmatrix} 1 & 1 & -2 & 1 & 0 & 0 & 0 & 2 \\ -2 & 0 & -2 & -3 & 1 & 0 & 0 & 2 \\ 5 & 0 & -1 & 1 & 0 & 1 & 0 & -4 \\ 8 & 0 & -1 & 2 & 0 & 0 & 1 & 4 \end{bmatrix}$$

This still hasn't made the bad row good. This time, there is a proper sign for improvement in column 3, and no choice of pivot row above it. In this situation we know we can make the bad row good by pivoting in the bad row itself, which we now do.

```
> E:=mulrow(C,3,-1):E=pivot(E,3,3);
```

$$E = \begin{bmatrix} -9 & 1 & 0 & -1 & 0 & -2 & 0 & 10 \\ -12 & 0 & 0 & -5 & 1 & -2 & 0 & 10 \\ -5 & 0 & 1 & -1 & 0 & -1 & 0 & 4 \\ 3 & 0 & 0 & 1 & 0 & -1 & 1 & 8 \end{bmatrix}$$

Now we see the situation we discussed at the start of the question. We are minimising the objective function, which thus has a proper sign for improvement in column 1. All the entries above it are negative, so we can make the objective function as large and negative as we wish.

***Solution*** 4.3:

a) We do the pivot suggested, and show the result below.

| Basis | ... | $\mathbf{a}_s$ | ... | $\mathbf{b}$ |
|---|---|---|---|---|
| $\vdots$ | | $\vdots$ | | $\vdots$ |
| $\mathbf{a}_{h_i}$ | ... | 0 | ... | $u_i - y_{is}\left(\frac{u_r}{y_{rs}}\right)$ |
| $\vdots$ | | $\vdots$ | | $\vdots$ |
| $\mathbf{a}_{h_r}$ | ... | 1 | ... | $\frac{u_r}{y_{rs}}$ |
| $\vdots$ | | $\vdots$ | | $\vdots$ |

Note that since $u_r > 0$ because the initial tableau was feasible, we must have $y_{rs} > 0$ in order that row $r$ be feasible afterwards. The other feasibility requirement is that $u_i - y_{is}\left(\frac{u_r}{y_{rs}}\right) > 0$. If $y_{is} < 0$, this will always hold. However if $y_{is} > 0$, then we must have

$$\frac{u_i}{y_{is}} \geq \frac{u_r}{y_{rs}}$$

and $r$ must be chosen to give the lowest such ratio among all rows $i$ for which $y_{is} > 0$.

b) To solve the given problem we first introduce slack variables in the first and last constraints, and an artificial variable, $x_5$ in the second constraint to get the following set of constraints

$$\begin{aligned} x_1 + x_3 + x_4 &= 4, \\ x_1 - 2x_2 - 3x_3 + x_5 &= 2, \\ x_1 - 3x_2 - x_3 + x_6 &= 1. \end{aligned}$$

Putting these into tableau form gives the following initial tableau.

```
> with(linalg):
> A:=matrix(4,3,[[1,0,1],[1,-2,-3],[1,-3,-1],[2,-1,4]]):
> B:= concat(A,diag(1,1,1,1),vector([4,2,1,0]));
```

$$B := \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 & 4 \\ 1 & -2 & -3 & 0 & 1 & 0 & 0 & 2 \\ 1 & -3 & -1 & 0 & 0 & 1 & 0 & 1 \\ 2 & -1 & 4 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Since column 5 corresponds to an artificial variable we first swap $\mathbf{a}_5$ from the basis, and then leave it out, since this ensures that $x_5 = 0$. It is natural to swap in $\mathbf{a}_1$ both because it is technically easier, and because it keeps row 2 as a good row.

```
> C:=pivot(B,2,1);
```

$$C := \begin{bmatrix} 0 & 2 & 4 & 1 & -1 & 0 & 0 & 2 \\ 1 & -2 & -3 & 0 & 1 & 0 & 0 & 2 \\ 0 & -1 & 2 & 0 & -1 & 1 & 0 & -1 \\ 0 & 3 & 10 & 0 & -2 & 0 & 1 & -4 \end{bmatrix}$$

Now row 3 is a bad row. We treat it as a subsidiary objective function and try to increase its value. Ignoring column 5 as we must, there is only one proper sign for improvement, in column 2 and only one positive pivot, in row 1.

```
> E:=pivot(C,1,2):E:=mulrow(E,1,1/2);
```

$$E := \begin{bmatrix} 0 & 1 & 2 & \frac{1}{2} & \frac{-1}{2} & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 4 \\ 0 & 0 & 4 & \frac{1}{2} & \frac{-3}{2} & 1 & 0 & 0 \\ 0 & 0 & 4 & \frac{-3}{2} & \frac{-1}{2} & 0 & 1 & -7 \end{bmatrix}$$

This is a minimising problem, and there is a proper sign for improvement in column 3. The lowest ratio (0) occurs in row 3.

```
> F:=pivot(E,3,3):mulrow(F,3,1/4);
```

$$\begin{bmatrix} 0 & 1 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{-1}{2} & 0 & 1 \\ 1 & 0 & 0 & \frac{7}{8} & \frac{3}{8} & \frac{-1}{4} & 0 & 4 \\ 0 & 0 & 1 & \frac{1}{8} & \frac{-3}{8} & \frac{1}{4} & 0 & 0 \\ 0 & 0 & 0 & -2 & 1 & -1 & 1 & -7 \end{bmatrix}$$

Doing the pivot does not improve the objective function, but now we can see the algorithm has terminated since there are no remaining proper signs for improvement. (Again we ignore the

artificial variable in column 5.) The minimum value of the objective function is $-7$, which occurs when $x_1 = 4$, $x_2 = 1$ and $x_3 = 0$.

***Solution*** 4.4: To solve the given problem we first introduce slack variables in the first and last constraints, and an artificial variable, $x_5$ in the second constraint to get the following set of constraints

$$\begin{aligned} x_1 + 3x_3 + x_4 &= 2, \\ 2x_1 - x_2 - x_3 + x_5 &= 3, \\ -4x_1 + x_2 + 2x_3 + x_6 &= -7. \end{aligned}$$

Putting these into tableau form with the objective function $x_7 - 6X_1 + x_2 - 12x_3 = 0$ gives the following initial tableau.

```
> with(linalg):
> A:=matrix(4,8,[[1,0,3,1,0,0,0,2],[2,-1,-1,0,1,0,0,3],
> [-4,1,2,0,0,1,0,-7],[-6,1,-12,0,0,0,1,0]]);
```

$$A := \begin{bmatrix} 1 & 0 & 3 & 1 & 0 & 0 & 0 & 2 \\ 2 & -1 & -1 & 0 & 1 & 0 & 0 & 3 \\ -4 & 1 & 2 & 0 & 0 & 1 & 0 & -7 \\ -6 & 1 & -12 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Since column 5 corresponds to an artificial variable we first swap $\mathbf{a}_5$ from the basis, and then leave it out, since this ensures that $x_5 = 0$. It is natural to swap in $\mathbf{a}_1$ because it keeps row 2 as a good row.

```
> B:=mulrow(A,2,1/2):B:=pivot(B,2,1);
```

$$B := \begin{bmatrix} 0 & \frac{1}{2} & \frac{7}{2} & 1 & \frac{-1}{2} & 0 & 0 & \frac{1}{2} \\ 1 & \frac{-1}{2} & \frac{-1}{2} & 0 & \frac{1}{2} & 0 & 0 & \frac{3}{2} \\ 0 & -1 & 0 & 0 & 2 & 1 & 0 & -1 \\ 0 & -2 & -15 & 0 & 3 & 0 & 1 & 9 \end{bmatrix}$$

Now row 3 is a bad row. We treat it as a subsidiary objective function and try to increase its value. Ignoring column 5 as we must, there is only one proper sign for improvement, in column 2 and only one positive pivot, in row 1.

```
> C:=mulrow(B,1,2):C:=pivot(C,1,2);
```

$$C := \begin{bmatrix} 0 & 1 & 7 & 2 & -1 & 0 & 0 & 1 \\ 1 & 0 & 3 & 1 & 0 & 0 & 0 & 2 \\ 0 & 0 & 7 & 2 & 1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 4 & 1 & 0 & 1 & 11 \end{bmatrix}$$

This is a minimising problem, and there is a proper sign for improvement in column 3. The lowest ratio (0) occurs in row 3.

```
> E:=pivot(C,3,4);
```

$$E := \begin{bmatrix} 0 & 1 & 0 & 0 & -2 & -1 & 0 & 1 \\ 1 & 0 & \frac{-1}{2} & 0 & \frac{-1}{2} & \frac{-1}{2} & 0 & 2 \\ 0 & 0 & 7 & 2 & 1 & 1 & 0 & 0 \\ 0 & 0 & -15 & 0 & -1 & -2 & 1 & 11 \end{bmatrix}$$

Doing the pivot does not improve the objective function, but now we can see the algorithm has terminated since there are no remaining proper signs for improvement. (Again we ignore the

artificial variable in column 5.) The minimum value of the objective function is 11, which occurs when $x_1 = 2$, $x_2 = 1$ and $x_3 = 0$.

There seems no good reason to eliminate the artificial variable by pivoting at a negative entry. For information here is the result doing it one way:

```
> F:=mulrow(A,2,-1):F:=pivot(F,2,2);
```

$$F := \begin{bmatrix} 1 & 0 & 3 & 1 & 0 & 0 & 0 & 2 \\ -2 & 1 & 1 & 0 & -1 & 0 & 0 & -3 \\ -2 & 0 & 1 & 0 & 1 & 1 & 0 & -4 \\ -4 & 0 & -13 & 0 & 1 & 0 & 1 & 3 \end{bmatrix}$$

```
> G:=pivot(F,1,1);
```

$$G := \begin{bmatrix} 1 & 0 & 3 & 1 & 0 & 0 & 0 & 2 \\ 0 & 1 & 7 & 2 & -1 & 0 & 0 & 1 \\ 0 & 0 & 7 & 2 & 1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 4 & 1 & 0 & 1 & 11 \end{bmatrix}$$

```
> H:=pivot(G,3,4);
```

$$H := \begin{bmatrix} 1 & 0 & \frac{-1}{2} & 0 & \frac{-1}{2} & \frac{-1}{2} & 0 & 2 \\ 0 & 1 & 0 & 0 & -2 & -1 & 0 & 1 \\ 0 & 0 & 7 & 2 & 1 & 1 & 0 & 0 \\ 0 & 0 & -15 & 0 & -1 & -2 & 1 & 11 \end{bmatrix}$$

Making a third choice gives one more step - but the initial pivot really doesn't look promising.

```
> J:=mulrow(A,2,-1):J:=pivot(J,2,3);
```

$$J := \begin{bmatrix} 7 & -3 & 0 & 1 & 3 & 0 & 0 & 11 \\ -2 & 1 & 1 & 0 & -1 & 0 & 0 & -3 \\ 0 & -1 & 0 & 0 & 2 & 1 & 0 & -1 \\ -30 & 13 & 0 & 0 & -12 & 0 & 1 & -36 \end{bmatrix}$$

```
> K:=mulrow(J,1,1/7):K:=pivot(K,1,1);
```

$$K := \begin{bmatrix} 1 & \frac{-3}{7} & 0 & \frac{1}{7} & \frac{3}{7} & 0 & 0 & \frac{11}{7} \\ 0 & \frac{1}{7} & 1 & \frac{2}{7} & \frac{-1}{7} & 0 & 0 & \frac{1}{7} \\ 0 & -1 & 0 & 0 & 2 & 1 & 0 & -1 \\ 0 & \frac{1}{7} & 0 & \frac{30}{7} & \frac{6}{7} & 0 & 1 & \frac{78}{7} \end{bmatrix}$$

```
> L:=mulrow(K,2,7):L:=pivot(L,2,2);
```

$$L := \begin{bmatrix} 1 & 0 & 3 & 1 & 0 & 0 & 0 & 2 \\ 0 & 1 & 7 & 2 & -1 & 0 & 0 & 1 \\ 0 & 0 & 7 & 2 & 1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 4 & 1 & 0 & 1 & 11 \end{bmatrix}$$

```
> M:=mulrow(L,3,1/2):M:=pivot(M,3,4);
```

$$M := \begin{bmatrix} 1 & 0 & \frac{-1}{2} & 0 & \frac{-1}{2} & \frac{-1}{2} & 0 & 2 \\ 0 & 1 & 0 & 0 & -2 & -1 & 0 & 1 \\ 0 & 0 & \frac{7}{2} & 1 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & -15 & 0 & -1 & -2 & 1 & 11 \end{bmatrix}$$

## Solutions for Questions 4 (page 64).

***Solution*** 5.1: We first rewrite the primal in "standard form" as: maximise $-6x_1 - 7x_2 - x_3 - 15x_4$, subject to $x_1, x_2, x_4, x_4 \geq 0$ and

$$\begin{aligned} -x_1 - x_2 + x_3 - 2x_4 &\leq -5, \\ x_1 + x_2 + 3x_4 &\leq 6, \\ -x_1 - x_2 - 3x_4 &\leq -6. \end{aligned}$$

Thus if we let $\mathbf{b} = (-5, 6, -6)^T$, $\mathbf{c} = (-6, -7, -1, -15)$ and

$$A = \begin{pmatrix} -1 & -1 & 1 & -2 \\ 1 & 1 & 0 & 3 \\ -1 & -1 & 0 & -3 \end{pmatrix},$$

the primal is: maximise $\mathbf{c}^T\mathbf{x}$ subject to $A\mathbf{x} \leq \mathbf{b}$ and $\mathbf{x} \geq \mathbf{0}$. The dual then becomes: minimise $\mathbf{b}^T\mathbf{w}$ subject to $A^T\mathbf{w} \geq \mathbf{c}$ and $\mathbf{w} \geq \mathbf{0}$. In terms of the definitions just given, the dual is thus:

minimise $-5w_1 + 6w_2 - 6w_3$, subject to $w_1, w_2, w_3 \geq 0$ and

$$\begin{aligned} -w_1 + w_2 - w_3 &\geq -6, \\ -w_1 + w_2 - w_3 &\geq -7, \\ w_1 &\geq -1, \\ -2w_1 + 3w_2 - 3w_3 &\geq -15. \end{aligned}$$

Note also that by writing $w_4 = w_2 - w_3$, we can replace all references to the non-negative variables $w_2$ and $w_3$, both in the objective function and the constraints, by the unrestricted variable $w_4$.

We are given in Question 4.1 that the primal has the optimal value 33. Note that this was for the given objective function; we changed the sign of this above, so for *this* primal, the optimal value was $-33$. It follows from the fundamental theorem of duality that the dual has an optimal solution with the same optimal value, namely $-33$.

***Solution*** 5.2:

a) The dual problem is to minimise $\mathbf{b}^T\mathbf{w}$ subject to $A^T\mathbf{w} \geq \mathbf{c}, \mathbf{w} \geq \mathbf{0}$. Now suppose that $\mathbf{x}$ and $\mathbf{w}$ are feasible solutions of the above primal and its dual. Note first that if $\mathbf{u} \leq \mathbf{v}$ and $\mathbf{x} \geq \mathbf{0}$ then $\mathbf{x}^T\mathbf{u} \leq \mathbf{x}^T\mathbf{v}$ since we first multiply an inequality by $x_j \geq 0$ and then add. Then, since $\mathbf{c} \leq A^T\mathbf{w}$, and $\mathbf{x} \geq \mathbf{0}$, we have

$$\mathbf{x}^T\mathbf{c} \leq \mathbf{x}^T A^T\mathbf{w} = (A\mathbf{x})^T\mathbf{w} \leq \mathbf{b}^T\mathbf{w}$$

where in the last inequality we have used the note above and the fact that $A\mathbf{x} \leq \mathbf{b}$. Since $\mathbf{x}^T\mathbf{c} = \mathbf{c}^T\mathbf{x}$, the result follows.

If $\mathbf{c}^T\mathbf{x} = \mathbf{b}^T\mathbf{w}$ then $\mathbf{x}$ and $\mathbf{w}$ are *optimal* solutions of the primal and dual. To see this, recall that for the primal problem, we are trying to maximise

$$\sum_{j=1}^{n} c_j x_j \leq \sum_{i=1}^{m} b_i w_i,$$

where the inequality is the result we have just proved. If the above inequality is actually an equality, then however we change any $x_j$ cannot increase the value of the objective function $\mathbf{c}^T\mathbf{x}$. And in the same way, however we change any $w_i$ cannot decrease the value of the objective function $\mathbf{b}^T\mathbf{w}$ of the dual problem. So both must already correspond to optimal solutions; indeed we see that both the primal and the dual problem have optimal solutions with the *same* optimal value.

b) The original problem was to maximise $z = 10x_1 + 20x_2 - 10x_3$ subject to constraints that $x_1 \geq 0$, $x_2 \geq 0$, $x_3 \geq 0$ and

$$\begin{aligned} 2x_1 + 3x_2 - x_3 &\leq 4 \\ -x_1 - 2x_2 &\leq 3 \\ -x_1 + x_2 - 2x_3 &\leq 1. \end{aligned}$$

The dual problem then becomes one of minimising $4w_1 + 3w_2 + w_3$ subject to constraints that $w_1 \geq 0$, $w_2 \geq 0$, $w_3 \geq 0$ and

$$\begin{aligned} 2w_1 - w_2 - w_3 &\leq 10 \\ 3w_1 - 2w_2 + w_3 &\leq 20 \\ -w_1 - 2w_3 &\leq -10. \end{aligned}$$

For completeness, the full tableau for the primal is given.

| Basis | $\mathbf{a}_1$ | $\mathbf{a}_2$ | $\mathbf{a}_3$ | $\mathbf{a}_4$ | $\mathbf{a}_5$ | $\mathbf{a}_6$ | $\mathbf{e}$ | $\mathbf{b}$ |
|---|---|---|---|---|---|---|---|---|
| $\mathbf{a}_4$ | 2 | 3 | -1 | 1 | 0 | 0 | 0 | 4 |
| $\mathbf{a}_5$ | -1 | -2 | 0 | 0 | 1 | 0 | 0 | 3 |
| $\mathbf{a}_6$ | -1 | $1^*$ | -2 | 0 | 0 | 1 | 0 | 1 |
| $\mathbf{e}$ | -10 | -20 | 10 | 0 | 0 | 0 | 1 | 0 |
| $\mathbf{a}_4$ | 5 | 0 | $5^*$ | 1 | 0 | -3 | 0 | 1 |
| $\mathbf{a}_5$ | -3 | 0 | -4 | 0 | 1 | 2 | 0 | 5 |
| $\mathbf{a}_2$ | -1 | 1 | -2 | 0 | 0 | 1 | 0 | 1 |
| $\mathbf{e}$ | -30 | 0 | -30 | 0 | 0 | 20 | 1 | 20 |
| $\mathbf{a}_3$ | 1 | 0 | 1 | 1/5 | 0 | -3/5 | 0 | 1/5 |
| $\mathbf{a}_5$ | 1 | 0 | 0 | 4/5 | 1 | -2/5 | 0 | 29/5 |
| $\mathbf{a}_2$ | 1 | 1 | 0 | 2/5 | 0 | -1/5 | 0 | 7/5 |
| $\mathbf{e}$ | 0 | 0 | 0 | 6 | 0 | 2 | 1 | 26 |

This gives an optimum value of 26 for the primal problem with solution $x_1 = 0$, $x_2 = 7/5$ and $x_3 = 1/5$. The corresponding dual problem has solution $w_1 = 6$, $w_2 = 0$ and $w_3 = 2$. Of course the optimal value remains at 26.

There was a choice of pivot in the second tableau; an equally acceptable choice is to pivot at row1, column1; doing so gives:-

| Basis | $\mathbf{a}_1$ | $\mathbf{a}_2$ | $\mathbf{a}_3$ | $\mathbf{a}_4$ | $\mathbf{a}_5$ | $\mathbf{a}_6$ | $\mathbf{e}$ | $\mathbf{b}$ |
|---|---|---|---|---|---|---|---|---|
| $\mathbf{a}_1$ | 1 | 0 | 1 | 1/5 | 0 | -3/5 | 0 | 1/5 |
| $\mathbf{a}_5$ | 0 | 0 | -1 | 3/5 | 1 | 1/5 | 0 | 28/5 |
| $\mathbf{a}_2$ | 0 | 1 | -1 | 1/5 | 0 | 2/5 | 0 | 6/5 |
| $\mathbf{e}$ | 0 | 0 | 0 | 6 | 0 | 2 | 1 | 26 |

which gives the same solution as the above for the dual, and of course the same optimum value of $-26$. The solution to the primal problem differs from the one obtained above, having $x_1 = 1/5$, $x_2 = 6/5$ and $x_3 = 0$; however the optimal value of course remains the same.

***Solution*** 5.3: Given a primal: maximise $\mathbf{c}^T\mathbf{x}$ subject to $\mathbf{Ax} \leq \mathbf{b}$ and $\mathbf{x} \geq \mathbf{0}$, the dual problem is: minimise $\mathbf{b}^T\mathbf{w}$ subject to $\mathbf{A}^T\mathbf{w} \geq \mathbf{c}$ and $\mathbf{w} \geq \mathbf{0}$. Rewriting the primal we have: maximise $-8x_1 - 5x_2 - 4x_3$ subject to the constraints $x_1 \geq 0$, $x_2 \geq 0$, $x_3 \geq 0$, and

$$\begin{aligned} -x_1 - 5x_2 + x_3 &\leq -1, \\ -x_1 + 2x_2 - x_3 &\leq 4. \end{aligned}$$

The dual problem is thus: minimise $-w_1 + 4w_2$ subject to the constraints $w_1 \geq 0$, $w_2 \geq 0$, and

$$\begin{aligned} -w_1 - w_2 &\geq -8, \\ -5w_1 + 2w_2 &\geq -5, \\ w_1 - w_2 &\geq -4. \end{aligned}$$

Of course both the requirement (to minimise), and the constraints, can be changed by multiplying by an appropriate factor.

If both the primal and the dual problems have feasible solutions, then both have optimal solutions with the same optimal value.

## Solutions for Questions 5 (page 73).

***Solution*** 6.1: The payoff matrix to Rowman is

| | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 0 | 1 | 2 |
| 1 | 1 | −2 | 3 |
| 2 | 2 | 3 | −4 |

where we write Columnman's choice along the top row, and Rowman's choice down the first column.

The minima along each row are 0, $-2$ and $-4$, with maximum 0, while the maxima in each column are 2, 3 and 3, with minimum 2. Since $2 > 0$, the game is not strictly determined.

We first increase the value of the game by 4 to ensure that the corresponding matrix has entries all of which are non-negative. The related linear programming problem that Columnman has to solve is then to maximise $x_1 + x_2 + x_3$ subject to:

$$\begin{aligned} 4x_1 + 5x_2 + 6x_3 &\leq 3, \\ 5x_1 + 2x_2 + 7x_3 &\leq 1, \\ 6x_1 + 7x_2 &\leq 1. \end{aligned}$$

Putting this into the related standard form gives the initial tableau:

| Basis | $\mathbf{a}_1$ | $\mathbf{a}_2$ | $\mathbf{a}_3$ | $\mathbf{a}_4$ | $\mathbf{a}_5$ | $\mathbf{a}_6$ | **e** | **b** |
|---|---|---|---|---|---|---|---|---|
| $\mathbf{a}_2$ | 4 | 5 | 6 | 1 | 0 | 0 | 0 | 1 |
| $\mathbf{a}_3$ | 5 | 2 | 7 | 0 | 1 | 0 | 0 | 1 |
| $\mathbf{a}_1$ | 6 | 7 | 0 | 0 | 0 | 1 | 0 | 1 |
| **e** | -1 | -1 | -1 | 0 | 0 | 0 | 1 | 0 |

Although not asked for, the derivation of the final tableau is as follows:

```
> with(linalg):
> A:=matrix([[4,5,6],[5,2,7],[6,7,0],[-1,-1,-1]]):
> b:=vector([1,1,1,0]):
> A1:=concat(A,diag(1,1,1,1),b);
```

$$A1 := \begin{bmatrix} 4 & 5 & 6 & 1 & 0 & 0 & 0 & 1 \\ 5 & 2 & 7 & 0 & 1 & 0 & 0 & 1 \\ 6 & 7 & 0 & 0 & 0 & 1 & 0 & 1 \\ -1 & -1 & -1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

```
> A2:=pivot(A1,3,1):A2:=mulrow(A2,3,1/6);
```

$$A2 := \begin{bmatrix} 0 & \frac{1}{3} & 6 & 1 & 0 & \frac{-2}{3} & 0 & \frac{1}{3} \\ 0 & \frac{-23}{6} & 7 & 0 & 1 & \frac{-5}{6} & 0 & \frac{1}{6} \\ 1 & \frac{7}{6} & 0 & 0 & 0 & \frac{1}{6} & 0 & \frac{1}{6} \\ 0 & \frac{1}{6} & -1 & 0 & 0 & \frac{1}{6} & 1 & \frac{1}{6} \end{bmatrix}$$

```
> A3:=pivot(A2,2,3):A3:=mulrow(A3,2,1/7);
```

$$A3 := \begin{bmatrix} 0 & \frac{76}{21} & 0 & 1 & \frac{-6}{7} & \frac{1}{21} & 0 & \frac{4}{21} \\ 0 & \frac{-23}{42} & 1 & 0 & \frac{1}{7} & \frac{-5}{42} & 0 & \frac{1}{42} \\ 1 & \frac{7}{6} & 0 & 0 & 0 & \frac{1}{6} & 0 & \frac{1}{6} \\ 0 & \frac{-8}{21} & 0 & 0 & \frac{1}{7} & \frac{1}{21} & 1 & \frac{4}{21} \end{bmatrix}$$

```
> A4:=pivot(A3,1,2):A4:=mulrow(A4,1,21/76);
```

$$A4 := \begin{bmatrix} 0 & 1 & 0 & \frac{21}{76} & \frac{-9}{38} & \frac{1}{76} & 0 & \frac{1}{19} \\ 0 & 0 & 1 & \frac{23}{152} & \frac{1}{76} & \frac{-17}{152} & 0 & \frac{1}{19} \\ 1 & 0 & 0 & \frac{-49}{152} & \frac{21}{76} & \frac{23}{152} & 0 & \frac{2}{19} \\ 0 & 0 & 0 & \frac{2}{19} & \frac{1}{19} & \frac{1}{19} & 1 & \frac{4}{19} \end{bmatrix}$$

Thus the optimal value of the linear programming problem is $z = 19/4$, and the optimal strategies for Rowman and Columnman are both

$$\frac{19}{4}\left(\frac{2}{19}, \frac{1}{19}, \frac{1}{19}\right) = \left(\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\right).$$

The value of the original game is $1/z - 4 = 3/4$.

***Solution*** 6.2: We label the 6 possible choices for $H$ in an obvious way with the symbols $R1C1$, $R1C2$, $R1C3$, $R2C1$, $R2C2$ and $R2C3$. Similarly we label the five choices of $S$ as $R1$, $R2$, $C1$, $C2$ and $C3$. We then describe all possible outcomes of the game in the following matrix, in which each row is labelled with a different choice by $H$ and each column with a different choice by $S$.

| | $R1C1$ | $R1C2$ | $R1C3$ | $R2C1$ | $R2C2$ | $R2C3$ | Min |
|---|---|---|---|---|---|---|---|
| $R1$ | 1 | 2 | 3 | −1 | −1 | −1 | −1 |
| $R2$ | −1 | −1 | −1 | 2 | 4 | 3 | −1 |
| $C1$ | 1 | −1 | −1 | 2 | −1 | −1 | −1 |
| $C2$ | −1 | 2 | −1 | −1 | 4 | −1 | −1 |
| $C3$ | −1 | −1 | 3 | −1 | −1 | 3 | −1 |
| Max | 1 | 2 | 3 | 2 | 4 | 3 | |

This matrix then becomes the payoff matrix for $S$. Around the matrix, the minimum in each row, and maximum in each column have been computed. We have

$$\alpha = \max_i \min_j a_{ij} \qquad \beta = \min_j \max_i a_{ij}.$$

Thus $\alpha = -1$ and $\beta = 1$ and the game is not strictly determined.

***Solution*** 6.3: Let $\mathbf{A} = [a_{ij}]$ and recall that if $\alpha = \max_i \min_j a_{ij}$ and $\beta = \min_j \max_i a_{ij}$, then $\alpha$ and $\beta$ are respectively the **lower** and **upper** values for the matrix game $\mathbf{A}$.

In our case,

$$\mathbf{A} = \begin{array}{c} \\ \\ \\ \\ \text{Max} \end{array} \begin{array}{c} \begin{array}{cc} & \end{array} \\ \begin{pmatrix} 4 & 2 \\ 8 & 6 \\ 2 & 4 \end{pmatrix} \\ \begin{array}{cc} 8 & 6 \end{array} \end{array} \begin{array}{c} \text{Min} \\ 2 \\ 6 \\ 2 \\ \\ \end{array} \quad \text{while} \quad \mathbf{B} = \begin{array}{c} \\ \\ \\ \\ \text{Max} \end{array} \begin{array}{c} \begin{array}{cc} & \end{array} \\ \begin{pmatrix} 2 & 6 \\ 8 & 4 \\ 4 & 2 \end{pmatrix} \\ \begin{array}{cc} 8 & 6 \end{array} \end{array} \begin{array}{c} \text{Min} \\ 2 \\ 4 \\ 2 \\ \\ \end{array}$$

and so for $\mathbf{A}$, we have $\alpha = \beta = 6$ and the game is strictly determined, while for $\mathbf{B}$ we have $\alpha = 4$, while $\beta = 6$.

We solve the game $\mathbf{B}$ using the simplex method, so we seek a $\mathbf{y}$ which is an optimum strategy for Columnman; ie we maximise $y_1 + y_2$ subject to $\mathbf{By} \leq \mathbf{1}$ and $y_1 \geq 0$, $y_2 \geq 0$.

The initial tableau is

```
> B:=matrix(4,2,[2,6,8,4,4,2,-1,-1]):
> B:=concat(B,diag(1,1,1,1),vector([1,1,1,0]));
```

$$B := \begin{bmatrix} 2 & 6 & 1 & 0 & 0 & 0 & 1 \\ 8 & 4 & 0 & 1 & 0 & 0 & 1 \\ 4 & 2 & 0 & 0 & 1 & 0 & 1 \\ -1 & -1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Following the simplex algorithm, we choose to pivot about the element 8 in column 1, since each of columns 1 and 2 have the proper sign for improvement. Pivoting gives the tableau

```
> C:=pivot(B,2,1):C:=mulrow(C,2,1/8);
```

$$C := \begin{bmatrix} 0 & 5 & 1 & \frac{-1}{4} & 0 & 0 & \frac{3}{4} \\ 1 & \frac{1}{2} & 0 & \frac{1}{8} & 0 & 0 & \frac{1}{8} \\ 0 & 0 & 0 & \frac{-1}{2} & 1 & 0 & \frac{1}{2} \\ 0 & \frac{-1}{2} & 0 & \frac{1}{8} & 0 & 1 & \frac{1}{8} \end{bmatrix}$$

Continuing, we see that column 2 has the proper sign for improvement, and we must pivot about 5, giving the tableau

```
> E:=pivot(C,1,2):E:=mulrow(E,1,1/5);
```

$$E := \begin{bmatrix} 0 & 1 & \frac{1}{5} & \frac{-1}{20} & 0 & 0 & \frac{3}{20} \\ 1 & 0 & \frac{-1}{10} & \frac{3}{20} & 0 & 0 & \frac{1}{20} \\ 0 & 0 & 0 & \frac{-1}{2} & 1 & 0 & \frac{1}{2} \\ 0 & 0 & \frac{1}{10} & \frac{1}{10} & 0 & 1 & \frac{1}{5} \end{bmatrix}$$

The optimum strategy is thus $\frac{5}{1}\left(\frac{1}{20}, \frac{3}{20}\right) = \left(\frac{1}{4}, \frac{3}{4}\right)$ for Columnman. The optimal strategy for Rowman is $\frac{5}{1}\left(\frac{1}{10}, \frac{1}{10}, 0\right) = \left(\frac{1}{2}, \frac{1}{2}, 0\right)$.

The expected payoff if both players use this strategy is thus

$$\frac{1}{4}\left(\frac{2}{2}+\frac{8}{2}\right)+\frac{3}{4}\left(\frac{6}{2}+\frac{4}{2}\right)=5.$$

The first game thus has a larger expected payoff to Rowman, and so is the logical choice.

***Solution*** 6.4: Let $\mathbf{A}=[a_{ij}]$ and recall that if $\alpha=\max_i \min_j a_{ij}$ and $\beta=\min_j \max_i a_{ij}$, then $\alpha$ and $\beta$ are respectively the **lower** and **upper** values for the matrix game $\mathbf{A}$.

In our case,

$$\begin{array}{cccc} & & & \text{Min} \\ \mathbf{A}= & \begin{pmatrix} 3 & 2 \\ 2 & 6 \\ 1 & 5 \end{pmatrix} & & \begin{matrix} 2 \\ 2 \\ 1 \end{matrix} \\ \text{Max} & 3 & 6 & \end{array}$$

and so $\alpha=2$, while $\beta=3$, and the game is not strictly determined. It is clear that row 3 will not figure in Rowman's solution, since whether Columnman chooses column 1 or column 2, Rowman is better off choosing row 2.

We solve the game $\mathbf{A}$ using the simplex method, so we seek a $\mathbf{y}$ which is an optimum strategy for Columnman; ie we maximise $y_1+y_2$ subject to $\mathbf{Ay}\le\mathbf{1}$ and $y_1\ge 0$, $y_2\ge 0$.

The initial tableau is

| Basis | $\mathbf{a}_1$ | $\mathbf{a}_2$ | $\mathbf{a}_3$ | $\mathbf{a}_4$ | $\mathbf{a}_5$ | **e** | **b** |
|---|---|---|---|---|---|---|---|
| $\mathbf{a}_3$ | 3 | 2 | 1 | 0 | 0 | 0 | 1 |
| $\mathbf{a}_4$ | 2 | 6* | 0 | 1 | 0 | 0 | 1 |
| $\mathbf{a}_5$ | 1 | 5 | 0 | 0 | 1 | 0 | 1 |
| **e** | -1 | -1 | 0 | 0 | 0 | 1 | 0 |

Following the simplex algorithm, we choose to pivot about the element indicated $6^*$, since each of columns 1 and 2 have the proper sign for improvement. Pivoting gives the tableau

| Basis | $\mathbf{a}_1$ | $\mathbf{a}_2$ | $\mathbf{a}_3$ | $\mathbf{a}_4$ | $\mathbf{a}_5$ | **e** | **b** |
|---|---|---|---|---|---|---|---|
| $\mathbf{a}_3$ | 7/3* | 0 | 1 | -1/3 | 0 | 0 | 2/3 |
| $\mathbf{a}_2$ | 1/3 | 1 | 0 | 1/6 | 0 | 0 | 1/6 |
| $\mathbf{a}_5$ | -2/3 | 0 | 0 | -5/6 | 1 | 0 | 1/6 |
| **e** | -2/3 | 0 | 0 | 1/6 | 0 | 1 | 1/6 |

Continuing, we see that column 1 has the proper sign for improvement, and we must pivot about $7/3^*$, giving the tableau

| Basis | $\mathbf{a}_1$ | $\mathbf{a}_2$ | $\mathbf{a}_3$ | $\mathbf{a}_4$ | $\mathbf{a}_5$ | **e** | **b** |
|---|---|---|---|---|---|---|---|
| $\mathbf{a}_1$ | 1 | 0 | 3/7 | -1/7 | 0 | 0 | 2/7 |
| $\mathbf{a}_2$ | 0 | 1 | -1/7 | 3/14 | 0 | 0 | 1/14 |
| $\mathbf{a}_5$ | 0 | 0 | 2/7 | -13/14 | 1 | 0 | 5/14 |
| **e** | 0 | 0 | 2/7 | 1/14 | 0 | 1 | 5/14 |

The optimum strategy is thus $\frac{14}{5}\left(\frac{4}{14},\frac{1}{14}\right)=\left(\frac{4}{5},\frac{1}{5}\right)$ for Columnman. The optimal strategy for Rowman is $\frac{14}{5}\left(\frac{2}{7},\frac{1}{14},0\right)=\left(\frac{4}{5},\frac{1}{5},0\right)$.

The expected payoff if both players use this strategy is thus

$$\frac{4}{5}\left(3.\frac{4}{5}+2.\frac{1}{5}\right)+\frac{1}{5}\left(2.\frac{4}{5}+6.\frac{1}{5}\right)=\frac{14}{5}.$$

Note also that this lies between $\alpha$ and $\beta$.

There is a choice above. The other way looks essentially equivalent:

```
> with (linalg):
> A:=matrix(4,7,[[3,2,1,0,0,0,1],[2,6,0,1,0,0,1],[1,5,0,0,1,0,1],[-1,-1
> ,0,0,0,1,0]]);
```

$$A := \begin{bmatrix} 3 & 2 & 1 & 0 & 0 & 0 & 1 \\ 2 & 6 & 0 & 1 & 0 & 0 & 1 \\ 1 & 5 & 0 & 0 & 1 & 0 & 1 \\ -1 & -1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

```
> E:=mulrow(A,1,1/A[1,1]):E:=pivot(E,1,1);
```

$$E := \begin{bmatrix} 1 & \frac{2}{3} & \frac{1}{3} & 0 & 0 & 0 & \frac{1}{3} \\ 0 & \frac{14}{3} & \frac{-2}{3} & 1 & 0 & 0 & \frac{1}{3} \\ 0 & \frac{13}{3} & \frac{-1}{3} & 0 & 1 & 0 & \frac{2}{3} \\ 0 & \frac{-1}{3} & \frac{1}{3} & 0 & 0 & 1 & \frac{1}{3} \end{bmatrix}$$

```
> F:=mulrow(E,2,3/14):F:=pivot(F,2,2);
```

$$F := \begin{bmatrix} 1 & 0 & \frac{3}{7} & \frac{-1}{7} & 0 & 0 & \frac{2}{7} \\ 0 & 1 & \frac{-1}{7} & \frac{3}{14} & 0 & 0 & \frac{1}{14} \\ 0 & 0 & \frac{2}{7} & \frac{-13}{14} & 1 & 0 & \frac{5}{14} \\ 0 & 0 & \frac{2}{7} & \frac{1}{14} & 0 & 1 & \frac{5}{14} \end{bmatrix}$$

## Solutions for Questions 6 (page 86).

***Solution*** 7.1: Suppose there are constants $\lambda_1, \lambda_2, \ldots, \lambda_k \in \mathbb{R}$ such that $\mathbf{x}_0 \in \mathbb{R}^n$ is an *unconstrained* optimum (max or min) of

$$f(\mathbf{x}) - \sum_{j=1}^{k} \lambda_j c_j(\mathbf{x})$$

and that in addition $c_j(\mathbf{x}_0) = 0$ for $1 \leq j \leq k$. Then $\mathbf{x}_0$ is a solution of the *constrained* optimization problem:

$$\text{maximise } f(\mathbf{x}) \text{ subject to } c_j(\mathbf{x}) = 0 \text{ for } (1 \leq j \leq k).$$

We illustrate this with the given example; let

$$L(\mathbf{x}) = L(\mathbf{x}, \lambda) = x_1^2 + \cdots + x_n^2 - \lambda(x_1 + \ldots x_n - 1)$$

We seek critical points; they occur when

$$\frac{\partial L}{\partial x_i} = 2x_i - \lambda = 0 \qquad \frac{\partial L}{\partial \lambda} = x_1 + \ldots x_n - 1 = 0.$$

There is a solution, when $x_i = \lambda/2$ and so $n\lambda = 2$; thus when $x_1 = \cdots = x_n = 1/n$. This solution is unique and is clearly a global minimum of $L(\mathbf{x}, \lambda)$, and hence of the given constrained optimization problem.

***Solution*** 7.2:

a) By Lagrange's principle, the required constrained maximum will occur at a local extremum of

$$F(x_1, x_2, x_3, \lambda_1\lambda_2) = x_1^2 + x_2^2 + x_3^2 - \lambda_1(x_1 + 2x_2 + x_3 - 4) + -\lambda_2(17 - x_1 + 2x_2 - 2x_3).$$

Then

$$\begin{aligned} 0 = \frac{\partial F}{\partial x_1} &= 2x_1 - \lambda_1 + \lambda_2, & x_1 &= \frac{1}{2}(\lambda_1 - \lambda_2), \\ 0 = \frac{\partial F}{\partial x_2} &= 2x_2 - 2\lambda_1 - 2\lambda_2, & x_2 &= \lambda_1 + \lambda_2, \\ 0 = \frac{\partial F}{\partial x_3} &= 2x_3 - \lambda_1 + 2\lambda_2 & x_3 &= \frac{1}{2}(\lambda_1 - 2\lambda_2). \end{aligned}$$

Using these values in the constraints (also obtained by taking the remaining partial derivatives), we have

$$0 = x_1 + 2x_2 + x_3 - 4 = \frac{1}{2}\lambda_1 - \frac{1}{2}\lambda_2 + 2\lambda_2 + 2\lambda_1 + \frac{1}{2}\lambda_1 - \lambda_2 - 4,$$

and thus $8 = \lambda_2 + 6\lambda_1$. The second constraint then gives

$$0 = x_1 - 2x_2 + 2x_3 - 17 = \frac{1}{2}(\lambda_1 - \lambda_2) - 2\lambda_2 - 2\lambda_1 + \lambda_1 - 2\lambda_2 - 17,$$

and thus $-\lambda_1 - 9\lambda_2 = 34$.

Solving this pair of equations gives $\lambda_1 = 2$ and $\lambda_2 = -4$, and then $x_1 = 3$, $x_2 = -2$ and $x_3 = 5$. To see this is a minimum, rather than a maximum, note that $x_2$ can be made arbitrarily large; this simply fixes $x_1$ and $x_3$.

b) If the objective function $f$ for a differentiable non-linear programming problem is convex on a domain $D$ and all the constraints are concave on $D$ then every solution which satisfies the Kuhn-Tucker conditions is necessarily a global minimum.

The Lagrangian is as above; this then gives the location of the critical points. Using these equations, The Kuhn-Tucker conditions are thus $\lambda_1 \geq 0$, $\lambda_2 \geq 0$ and:-

$$\begin{aligned} 2x_1 - \lambda_1 + \lambda_2 &= 0, \\ 2x_2 - 2\lambda_1 - 2\lambda_2 &= 0, \\ 2x_3 - \lambda_1 + 2\lambda_2 &= 0, \\ x_1 + 2x_2 + x_3 - 4 &\geq 0, \\ 17 - x_1 - 2x_2 - 2x_3 &\geq 0, \\ \lambda_1(x_1 + 2x_2 + x_3 - 4) &= 0, \\ \lambda_2(17 - x_1 + 2x_2 - 2x_3) &= 0. \end{aligned}$$

Of course, if $\lambda_1 \neq 0$ and $\lambda_2 \neq 0$ each constraint is tight, and we have the same solution as before. We may exclude it because that solution gave $\lambda_2 = -4$, which fails the KT conditions.

If $\lambda_1 = 0$ and $\lambda_2 = 0$, the only solution is $x_1 = 0$, $x_2 = 0$ and $x_3 = 0$, which is not feasible since $x_1 + 2x_2 + x_3 \geq 4$.

If $\lambda_1 = 0$, by KT and the above we must have $\lambda_2 > 0$ and $17 = x_1 + 2x_2 - 2x_3$. Since $x_1 = -\lambda_2/2$, $x_2 = \lambda_2$ and $x_3 = -\lambda_2$, we see that $x_1 - 2x_2 + 2x_3 < 0$, and again we reach a contradiction. The remaining possibility is that $\lambda_1 > 0$ and $\lambda_2 = 0$. In this case, $x_1 = \lambda_1/2$, $x_2 = \lambda_1$ and $x_3 = \lambda_1/2$. Since by KT we must have $x_1 + 2x_2 + x_3 = 4$,

$$\frac{1}{2}\lambda_1 + 2\lambda_1 + \frac{1}{2}\lambda_1 = 4 \quad \text{so that} \quad \lambda_1 = \frac{4}{3}.$$

The corresponding solution is $\left(\frac{2}{3}, \frac{4}{3}, \frac{2}{3}\right)$, and the minimum value attained is $\frac{24}{9}$. Note also that the constraints are indeed satsified at this point. Finally, observe that since the objective function is convex, and the constraints are linear, and hence concave, the solution is a global minimum.

***Solution*** 7.3: We show below the two constraints and two different positions of the objective function

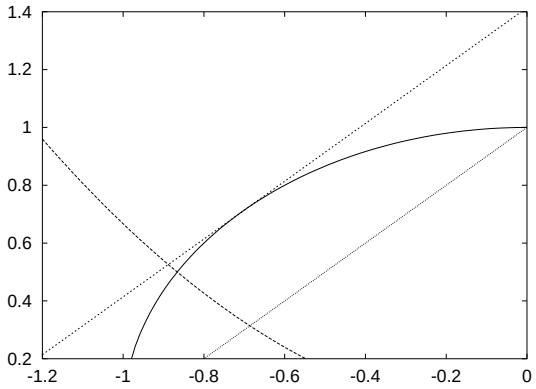


Figure A.1: The feasible region is above the parabola and below the circle.

The objective function becomes smaller as the intercept on the $x_2$ - axis increase; in other words, as the line moves to the northwest!

The Lagrangian is

$$L(x_1, x_2, \lambda, \mu) = x_1 - x_2 - \lambda(1 - x_1^2 - x_2^2) - \mu(3x_2 - 2x_1^2).$$

Using these equations, The Kuhn-Tucker conditions are thus $\lambda \geq 0$, $\mu \geq 0$ and

$$\begin{aligned} 1 + \lambda 2x_1 + 4\mu x_1 &= 0, & -1 + 2\lambda x_2 - 3\mu &= 0, \\ 1 - x_1^2 - x_2^2 &\geq 0, & 2x_1^2 - 3x_2 &\geq 0, \\ \lambda(1 - x_1^2 - x_2^2) &= 0, & \mu(2x_1^2 - 3x_2) &= 0. \end{aligned}$$

and solutions to these equations give the local minimisers — strictly unless a certain tangent condition is non-degenerate, as it is in this case.

The first equation shows that $\lambda = \mu = 0$ does not give a solution; thus at least one of the constraints is tight.

If $\lambda = 0$, the second equation gives $\mu = -1/3$, which contradicts the fact that $\mu \geq 0$.

If $\lambda > 0$ and $\mu > 0$ so both constraints are tight, $x_1$ and $x_2$ satisfy

$$x_1^2 + x_2^2 = 1, \qquad 2x_1 = 3x_2.$$

Eliminating $x_1$ gives

$$3x_2 + 2x_2^2 - 2 = 0 = (2x_2 - 1)(x_2 + 2)$$

and $x_2 = 1/2$, since the other solution, $x_2 = -2$ does not give a real solution for $x_1$. This gives $x_1 = \pm\sqrt{3}/2$ and a minimum value for $z$ of $-(\sqrt{3} - 1)/2$.

Finally we consider the case $\mu = 0$, $\lambda > 0$ so $x_1^2 + x_2^2 = 1$. From our diagram we expect this to give the global minimum. Since $\mu = 0$, the first equation gives $1 + 2\lambda x_1 = 0$, and $x_1 = -1/(2\lambda)$, while the second equation gives $x_2 = 1/(2\lambda)$. Since $\lambda > 0$, the "circle" constraint, that $x_1^2 + x_2^2 \leq 1$ is tight (ie an equality) and $2\lambda^2 = 1$. This corresponds to the solution $x_1 = -1/\sqrt{2}$, $x_2 = 1/\sqrt{2}$ and an objective value of $z = -\sqrt{2}$ which is thus the minimum.

## Solutions for Questions 7 (page 104).

***Solution*** 9.1: This is a brief sketch of the expected answer.

The aim in a genetic algorithm is to encode possible solutions in strings. For the example given, it is reasonable to use a binary string, which represents the particular value of $n$ which is the solution encoded by the string. Associated with each string is a notion of *fitness*, indicating how good the string is as a solution of the problem. Again in the example, the fitness function is clear; it is to be the value of the objective function at the number represented by the stirng. Strings with a greater fitness represent better attempts at the required maximisation.

An initial population of strings is chosen at random, and the fitness of each string is calculated. Pairs of strings are then selected for mating, and offspring collected to form the next generation. Selection for mating is biased in such a way that fit strings are most likely to be selected. For simplicity, we assume that each pair of "parent" strings produces two "offstrings" (groan). Mating continues until (for convenience) the new population is the same size as the old one, when a single generation is complete.

Mating itself consists of interchanging "genetic material". One or more cuts points are selected, each parent's string is cut at these points, and crossover occurs, so that each offstring is build from a mixture of both parents' strings. One further process is necessary, that of a random mutation, in which each bit in a child string is considered for flipping. This is allowed to occcur with a very low probability as a way of introducing useful parts of the string spectrum which have otherwise never been explored, or have died out.

In operation, the population is allowed to evolve over a large number (perhaps 1000) of generations, and the fitest individual in the final population is taken as the solution. A variant stores the best individual encountered during the evolution, perhaps even allowing it to be immortal, at least until a fitter individual is encountered.

Two additional process should be mentioned. One of these is the code used initially. Although binary code is suggested above, a Gray coding is preferable, so that adjacent values of $n$ are coded with strings which are always very similar, as oppposed to the "Hamming cliffs" which occur eg between 15 and 16 in standard binary encoding. The second process, called "fitness scaling" is needed to avoid the population between swamped by a few very fit individuals, to the exclusion of other genetic material. The fitness of fit individuals is often limited to some maximum value, perhaps twice that of the average fitness, in order to allow a more varied next generation.

***Solution*** 9.2:

a) Given the problem, in which we have to choose a binary state of each of a finite number of gates, it is clear how we can encode a program as a binary string. We are given a real number $x$, say, which measures the difference between the output from the program and the desired output; we take the fitness of the corresponding string to be $1/x$; strings with a greater fitness represent better attempts at the required program.

An initial population of strings is chosen at random, and the fitness of each string is calculated. Pairs of strings are then selected for mating, and offspring collected to form the next generation. Selection for mating is biased in such a way that fit strings are most likely to be selected. For simplicity, we assume that each pair of "parent" strings produces two "offstrings" (groan). Mating continues until (for convenience) the new population is the same size as the old one, when a single generation is complete.

Mating itself consists of interchanging "genetic material". One or more cuts points are selected, each parent's string is cut at these points, and crossover occurs, so that each offstring is build from a mixture of both parents' strings. One further process is necessary, that of a random mutation, in which each bit in a child string is considered for flipping. This is allowed to occur with a very low probability as a way of introducing useful parts of the string spectrum which have otherwise never been explored, or have died out.

In operation, the population is allowed to evolve over a large number (perhaps 1000) of generations, and the fittest individual in the final population is taken as the solution. A variant stores the best individual encountered during the evolution, perhaps even allowing it to be immortal, at least until a fitter individual is encountered. A process called called "fitness scaling" may be needed to avoid the population between swamped by a few very fit individuals, to the exclusion of other genetic material. The fitness of fit individuals is often limited to some maximum value, perhaps twice that of the average fitness, in order to allow a more varied next generation.

In case the example seems fanciful, I saw it *running* in Sussex; it was also featured in recent EPSRC publicity.

***Solution*** 9.3: Linear optimisation methods can be applied to an objective function which is a linear (strictly affine) function, and for which the constraints are also linear or affine. An algorithm is then available for converting such problems into a common "tableau" form. The simplex method can then be applied, either to move from one feasible solution to a better one, or in the two-phase case, to move towards a feasible solution in the first case, from which point the one-phase algorithm can be applied. The algorithm can terminate in one of three ways; there may be no feasible solution, in which case the two-phase algorithm terminate before it finds one; or an unbounded one parameter familiy of solutions will be found, each better than the previous one, or termination will occur at an optimal solution.

Typical problems are like the wine blending one above, or others, such as the transport problem, in which the linear and feasibility assumptions are natural.

None linear problems have a much wider range of applicability, but can be thought of as optimising a (not necessarily) linear objective function subject to constraints which are themselves are not necessarily linear. One such methodology is a genetic algorithm, in which different solutions are coded in (binary) strings, thought of a genes, and for which the objective function, applied to a binary string, is considered to measure the fitness of the string. A population of strings is chosen at random, and allowed to evolve, by processes including the natural selection of fit strings, mating of strings to include exchange of substrings, and a low-probability mutation process which aims to improve genetic diversity. The effcet of such an algorithm is to explore the whole solution space more efficiently than by exhaustive search.

Typical problems here are very much more divers, from simle non-linear optimisation, to generating rules for efficient game playing.

In general linear methods will be very much faster than non-linear ones, although with a much more restricted problem domain. However there are specialised non-linear algorithms which are themselves very efficient, when they embody much knowledge about the specific problem they are trying to solve. More general algorthms, such as genetic and simulated annealing ones have a wide

applicability; although they are not as good as specific algorithms for a given problem, they can often be applied rapidly to new types of problems.

If a problem was suitable for both a linear and non-linear method to be applied, then the problem is necessarily a linear one, and as such the linear methods are necessarily the most efficient, being designed for the problem. Some non-linear methods may do well, particularly calculus based ones, which make the assumption that the objective function and constraints are locally linear.

***Solution*** 9.4: For nearly linear problems, typically those that are convex, some variant of hill climbing can be used, moving in the direction of the largest rate of change of the objective function, and again having a guarantee of convergence. This could converge quickly, but with an ill conditioned objective function, could be much slower. For a more general problem, there are many less traditional methods, such as, for example a genetic algorithm, or simulated annealing, in which less structure is needed and a good approximation to the extremum is likely. although exhaustive search is a theoretical possibility, it is ruled out on performance grounds; in any case such a method is very slow compared with those for which more is known about the objective function.

If both linear and non-linear methods are applicable to the same problem, the linear one would be superior, being in general both more effective and faster. In fact some non-linear methods can reduce to the corresponding linear ones in a linear situation, but the purpose built methods will still be much more efficient; for example the simplex algorithm has a very simple accurate stopping rule.

***Solution*** 9.5:

a) One variant of the Travelling Salesman Problem is:

> A salesman has to visit $n$ cities and return to his city of origin. Each city has to be visited exactly once, and the cost of the journey between each pair of cities is known. The problem is to do the tour at minimum total cost.

We show the calculation below. We have chosen $A$ as the starting point and listed all possible tours; there is a unique tour with the minimum length of 26, namely $ACDB$.

| Tour | Step 1 | Step 2 | Step 3 | Step 4 | Total |
|---|---|---|---|---|---|
| A B C D | 4 | 6 | 10 | 11 | 31 |
| A B D C | 4 | 8 | 9 | 7 | 28 |
| A C B D | 6 | 10 | 8 | 11 | 35 |
| A C D B | 6 | 10 | 7 | 3 | 26 |
| A D B C | 12 | 7 | 6 | 7 | 32 |
| A D C B | 12 | 9 | 10 | 3 | 34 |

b) The two essential ingredients of a genetic algorithm are the availability of a natural representation of the problem and its constraints, which leads to suitable genetic operations, and the availability of a useful fitness function. In this problem, the fitness function is clear; two tours can be evaluated simply by looking at the total cost of each tour; in contrast there is no natural representation. which describes the problem.

Of course a naive binary representation of each tour is available, in which each city is numbered, and a gene is simply an ordered list of city numbers. This representation does not however lend itself to useful genetic operations; a naive mating may not even produce a tour (some city numbers may be invalid), and is unlikely to be a valid tour with no repeat cities.

We are thus led to seek alternative (non-binary) representations which are adapted to this particular problem. Here are some possible representations.

- the ordinal representation, in which the tour $(1, 2, 4, 3, 8, 5, 9, 6, 7)$ is represented as the list $(1, 1, 2, 1, 4, 1, 3, 1, 1)$ in which the code refers to the position of the next city of the list of unused cities.

- the path representation, which was the natural one we used above; and
- the adjacency representations in which the tour $(1, 2, 4, 3, 8, 5, 9, 6, 7)$ in the path representation is given as $(2, 4, 8, 3, 9, 7, 1, 5, 6)$, indicating by the entry in position $k$ that the tour goes from $k$ to this new city.

Even when cross-over is defined, it does not lead to natural "genetic evolution" and so it is necessary to have much more specialised crossover operators, in which the naive cross-over is "repaired".

c) We simply use the obvious path representation described above; thus each potential tour can be described by a re-arrangement $\mathbf{i}$ of the city labels.

We try to generate "intelligent" changes, and indeed this is at the heart of the method. Here are two possibilities:

- a section of the tour is cut out and replaced by one running the other way round; or
- the section is replaced by a random re-arrangement of that section.

The objective function is again the one given above. We choose the "temperature" parameter $T$ so that initially it will allow such changes with high probability. Then decrease $T$ by (say) 10% each time as part of the cooling schedule, and run the algorithm until, for example, either

- a total of $100N$ reconfigurations have occurred; or
- a total of $10N$ successful reconfigurations have occurred.

Next, decrease the temperature by 10% and repeat the calculations. Continue in this way until there is so little change that the system seems to have converged.

# Bibliography


Char, B. W., Geddes, K. O., Gonnet, G. H., Leong, B. L., Monagan, M. B. & Watt, S. M. (1992), *First Leaves: A Tutorial Introduction to Maple V*, Springer-Verlag.

Goldberg, D. E. (1989), *Genetic algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley. pp412.

Hartley, R. (1985), *Linear and Nonlinear Programming: An Introduction to Linear Methods in Mathematical Programming*, Ellis Horwood.

Kaplan, W. (1999), *Maxima and Minima with Applications*, John Wiley and Sons Inc.

Kolman, B. & Beck, R. E. (1995), *Elementary Linear Programming with Applications*, second edn, Academic Press.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A. & Teller, E. (1953), 'Simulated annealing', *Journal of Chemical Physics* **21**, 1087–1092.

Michalewicz, Z. (1996), *Genetic Algorithms + Data Structures = Evolution Programs*, third edn, Springer.

Michalewicz, Z. & Fogel, D. B. (2000), *How to Solve It; Modern Heuristics*, Springer.

Mitchell, M. (1996), *An Introduction to Genetic Algorithms*, The MIT Press.

Ngo, T. & Bhadkamkar, N. (1998), The cocktail party effect, *in* 'New Scientist', number 2159, Reed Business Information, pp. 51 – 52.

Press, W. H., Flannery, B. P., Teukolsky, S. A. & Vetterling, W. T. (1992), *Numerical recipes in C. The Art of Scientific Computing*, second edn, Cambridge University Press.

Smythe, W. R. & Johnson, L. A. (1966), *Introduction to Linear Programming with Applications*, Prentice-Hall.

Winston, W. L. (1995), *Introduction to Mathematical Programming: Applications and Algorithms*, second edn, Duxbury.

## Index Entries